Layered Image Motion with Explicit Occlusions, Temporal Consistency, and Depth Ordering

Deqing Sun, Erik B. Sudderth, and Michael J. Black
Department of Computer Science, Brown University
{dqsun,sudderth,black}@cs.brown.edu

Abstract

Layered models are a powerful way of describing natural scenes containing smooth surfaces that may overlap and occlude each other. For image motion estimation, such models have a long history but have not achieved the wide use or accuracy of non-layered methods. We present a new probabilistic model of optical flow in layers that addresses many of the shortcomings of previous approaches. In particular, we define a probabilistic graphical model that explicitly captures: 1) occlusions and disocclusions; 2) depth ordering of the layers; 3) temporal consistency of the layer segmentation. Additionally the optical flow in each layer is modeled by a combination of a parametric model and a smooth deviation based on an MRF with a robust spatial prior; the resulting model allows *roughness in layers*. Finally, a key contribution is the formulation of the layers using an image-dependent hidden field prior based on recent models for static scene segmentation. The method achieves state-of-the-art results on the Middlebury benchmark and produces meaningful scene segmentations as well as detected occlusion regions.

1 Introduction

Layered models of scenes offer significant benefits for optical flow estimation [8, 11, 25]. Splitting the scene into layers enables the motion in each layer to be defined more simply, and the estimation of motion boundaries to be separated from the problem of smooth flow estimation. Layered models also make reasoning about occlusion relationships easier. In practice, however, none of the current top performing optical flow methods use a layered approach [2]. The most accurate approaches are single-layered, and instead use some form of robust smoothness assumption to cope with flow discontinuities [5]. This paper formulates a new probabilistic, layered motion model that addresses the key problems of previous layered approaches. At the time of writing, it achieves the lowest average error of all tested approaches on the Middlebury optical flow benchmark [2]. In particular, the accuracy at occlusion boundaries is significantly better than previous methods. By segmenting the observed scene, our model also identifies occluded and disoccluded regions.

Layered models provide a segmentation of the scene and this segmentation, because it corresponds to scene structure, should persist over time. However, this persistence is not a benefit if one is only computing flow between two frames; this is one reason that multi-layer models have not surpassed their single-layer competitors on two-frame benchmarks. Without loss of generality, here we use three-frame sequences to illustrate our method. In practice, these three frames can be constructed from an image pair by computing both the forward and backward flow. The key is that this gives two segmentations of the scene, one at each time instant, both of which must be consistent with the flow. We formulate this *temporal layer consistency* probabilistically. Note that the assumption of temporal layer consistency is much more realistic than previous assumptions of temporal motion consistency [4]; while the scene motion can change rapidly, scene structure persists.

One of the main motivations for layered models is that, conditioned on the segmentation into layers, each layer can employ affine, planar, or other strong models of optical flow. By applying a single smooth motion across the entire layer, these models combine information over long distances and interpolate behind occlusions. Such rigid parametric assumptions, however, are too restrictive for real scenes. Instead one can model the flow within each layer as smoothly varying [26]. While the resulting model is more flexible than traditional parametric models, we find that it is still not as accurate as robust single-layer models. Consequently, we formulate a hybrid model that combines a base affine motion with a robust Markov random field (MRF) model of *deformations* from affine [6]. This *roughness in layers* model, which is similar in spirit to work on plane+parallax [10, 14, 19], encourages smooth flow within layers but allows significant local deviations.

Because layers are temporally persistent, it is also possible to reason about their relative depth ordering. In general, reliable recovery of depth order requires three or more frames. Our probabilistic formulation explicitly orders layers by depth, and we show that the correct order typically produces more probable (lower energy) solutions. This also allows explicit reasoning about occlusions, which our model predicts at locations where the layer segmentations for consecutive frames disagree.

Many previous layered approaches are not truly "layered": while they segment the image into multiple regions with distinct motions, they do not model what is in front of what. For example, widely used MRF models [27] encourage neighboring pixels to occupy the same region, but do not capture relationships between regions. In contrast, building on recent state-of-the-art results in static scene segmentation [21], our model determines layer support via an ordered sequence of occluding binary masks. These binary masks are generated by thresholding a series of random, continuous functions. This approach uses image-dependent Gaussian random field priors and favors partitions which accurately match the statistics of real scenes [21]. Moreover, the continuous layer support functions play a key role in accurately modeling temporal layer consistency. The resulting model produces accurate layer segmentations that improve flow accuracy at occlusion boundaries, and recover meaningful scene structure.

As summarized in Figure 1, our method is based on a principled, probabilistic generative model for image sequences. By combining recent advances in dense flow estimation and natural image segmentation, we develop an algorithm that simultaneously estimates accurate flow fields, detects occlusions and disocclusions, and recovers the layered structure of realistic scenes.

2 Previous Work

Layered approaches to motion estimation have long been seen as elegant and promising, since spatial smoothness is separated from the modeling of discontinuities and occlusions. Darrell and Pentland [7, 8] provide the first full approach that incorporates a Bayesian model, "support maps" for segmentation, and robust statistics. Wang and Adelson [25] clearly motivate layered models of image sequences, while Jepson and Black [11] formalize the problem using probabilistic mixture models. A full review of more recent methods is beyond our scope [1, 3, 12, 13, 16, 17, 20, 24, 27, 29].

Early methods, which use simple parametric models of image motion within layers, are not highly accurate. Observing that rigid parametric models are too restrictive for real scenes, Weiss [26] uses a more flexible Gaussian process to describe the motion within each layer. Even using modern implementation methods [22] this approach does not achieve state-of-the-art results. Allocating a separate layer for every small surface discontinuity is impractical and fails to capture important global scene structure. Our approach, which allows "roughness" within layers rather than "smoothness," provides a compromise that captures coarse scene structure as well as fine within-layer details.

One key advantage of layered models is their ability to realistically model occlusion boundaries. To do this properly, however, one must know the relative depth order of the surfaces. Performing inference over the combinatorial range of possible occlusion relationships is challenging and, consequently, only a few layered flow models explicitly encode relative depth [12, 30]. Recent work revisits the layered model to handle occlusions [9], but does not explicitly model the layer ordering or achieve state-of-the-art performance on the Middlebury benchmark. While most current optical flow methods are "two-frame," layered methods naturally extend to longer sequences [12, 29, 30].

Layered models all have some way of making either a hard or soft assignment of pixels to layers. Weiss and Adelson [27] introduce spatial coherence to these layer assignments using a spatial MRF

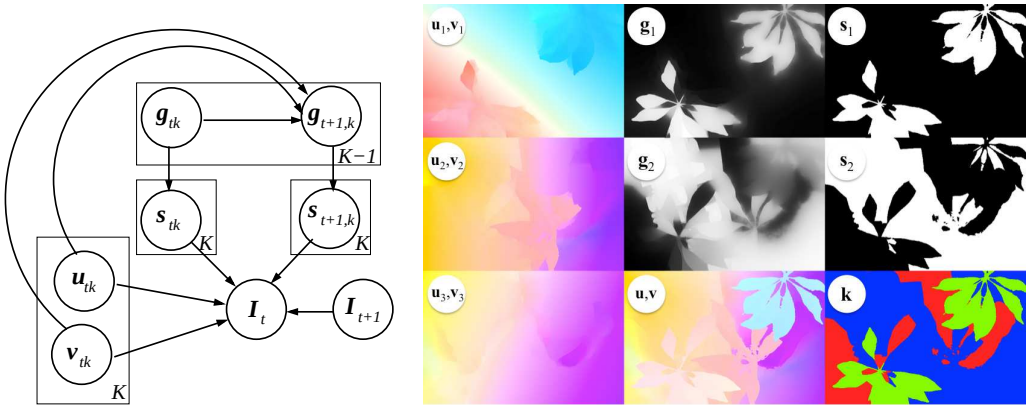


Figure 1: *Left:* Graphical representation for the proposed layered model. *Right:* Illustration of variables from the graphical model for the "Schefflera" sequence. Labeled sub-images correspond to nodes in the graph. The left column shows the flow fields for three layers, color coded as in [2]. The $\mathbf{g}$ and $\mathbf{s}$ images illustrate the reasoning about layer ownership (see text). The composite flow field $(\mathbf{u}, \mathbf{v})$ and layer labels $(\mathbf{k})$ are also shown.

model. However, the Ising/Potts MRF they employ assigns low probability to typical segmentations of natural scenes [15]. Adapting recent work on static image segmentation by Sudderth and Jordan [21], we instead generate spatially coherent, ordered layers by thresholding a series of random continuous functions. As in the single-image case, this approach realistically models the size and shape properties of real scenes. For motion estimation there are additional advantages: it allows accurate reasoning about occlusion relationships and modeling of temporal layer consistency.

3 A Layered Motion Model

Building on this long sequence of prior work, our generative model of layered image motion is summarized in Figure 1. Below we describe how the generative model captures piecewise smooth deviation of the layer motion from parametric models (Sec. 3.1), depth ordering and temporal consistency of layers (Sec. 3.2), and regions of occlusion and disocclusion (Sec. 3.3).

3.1 Roughness in Layers

Our approach is inspired by Weiss's model of smoothness in layers [26]. Given a sequence of images $\mathbf{I}_t, 1 \leq t \leq T$, we model the evolution from the current frame $\mathbf{I}_t$, to the subsequent frame $\mathbf{I}_{t+1}$, via K locally smooth, but potentially globally complex, flow fields. Let $\mathbf{u}_{tk}$ and $\mathbf{v}_{tk}$ denote the horizontal and vertical flow fields, respectively, for layer k at time t. The corresponding flow vector for pixel (i, j) is then denoted by $(u_{tk}^{ij}, v_{tk}^{ij})$.

Each layer's flow field is drawn from a distribution chosen to encourage piecewise smooth motion. For example, a pairwise Markov random field (MRF) would model the horizontal flow field as

$$p(\mathbf{u}_{tk}) \propto \exp\{-E_{\text{mrf}}(\mathbf{u}_{tk})\} = \exp\left\{-\frac{1}{2}\sum_{(i,j)}\sum_{(i',j')\in\Gamma(i,j)}\rho_s(u_{tk}^{ij} - u_{tk}^{i'j'})\right\}. \quad (1)$$

Here, $\Gamma(i, j)$ is the set of neighbors of pixel (i, j), often its four nearest neighbors. The potential $\rho_s(\cdot)$ is some robust function [5] that encourages smoothness, but allows occasional significant deviations from it. The vertical flow field $\mathbf{v}_{tk}$ can then be modeled via an independent MRF prior as in Eq. (1), as justified by the statistics of natural flow fields [18].

While such MRF priors are flexible, they capture very little dependence between pixels separated by even moderate image distances. In contrast, real scenes exhibit coherent motion over large scales, due to the motion of (partially) rigid objects in the world. To capture this, we associate an affine (or planar) motion model, with parameters θ_{tk}, to each layer k. We then use an MRF to allow piecewise smooth *deformations* from the globally rigid assumptions of affine motion:

$$E_{\text{aff}}(\mathbf{u}_{tk}, \theta_{tk}) = \frac{1}{2}\sum_{(i,j)}\sum_{(i',j')\in\Gamma(i,j)}\rho_s\Big((u_{tk}^{ij} - \bar{u}_{\theta_{tk}}^{ij}) - (u_{tk}^{i'j'} - \bar{u}_{\theta_{tk}}^{i'j'})\Big). \quad (2)$$

Here, $\bar{u}^{ij}_{\theta_{tk}}$ denotes the horizontal motion predicted for pixel (i,j) by an affine model with parameters θ_{tk}. Unlike classical models that assume layers are globally well fit by a single affine motion [6, 25], this prior allows significant, locally smooth deviations from rigidity. Unlike the basic smoothness prior of Eq. (1), this semiparametric construction allows effective global reasoning about non-contiguous segments of partially occluded objects. More sophisticated flow deformation priors may also be used, such as those based on robust non-local terms [22, 28].

3.2 Layer Support and Spatial Contiguity

The support for whether or not a pixel belongs to a given layer k is defined using a hidden random field $\mathbf{g}_k$. We associate each of the first $K-1$ layers at time t with a random continuous function $\mathbf{g}_{tk}$, defined over the same domain as the image. This hidden support field is illustrated in Figure 1.

We assume a single, unique layer is observable at each location and that the observed motion of that pixel is determined by its assigned layer. Analogous to level set representations, the discrete support of each layer is determined by thresholding $\mathbf{g}_{tk}$: pixel (i,j) is considered visible when $g_{tk}(i,j) \geq 0$. Let $s_{tk}(i,j)$ equal one if layer k is visible at pixel (i,j), and zero otherwise; note that $\sum_k s_{tk}(i,j) = 1$. For pixels (i,j) for which $g_{tk}(i,j) < 0$, we necessarily have $s_{tk}(i,j) = 0$.

We define the layers to be ordered with respect to the camera, so that layer k occludes layers $k' > k$. Given the full set of support functions $\mathbf{g}_{tk}$, the unique layer k^{ij}_{t*} for which $s_{tk^{ij}_{t*}}(i,j) = 1$ is then

$$k^{ij}_{t*} = \min\left(\{k \mid 1 \leq k \leq K-1, g_{tk}(i,j) \geq 0\} \cup \{K\}\right). \tag{3}$$

Note that layer K is essentially a background layer that captures all pixels not assigned to the first $K-1$ layers. For this reason, only $K-1$ hidden fields $\mathbf{g}_{tk}$ are needed (see Figure 1).

Our use of thresholded, random continuous functions to define layer support is partially motivated by known shortcomings of discrete Ising/Potts MRF models for image partitions [15]. They also provide a convenient framework for modeling the temporal and spatial coherence observed in real motion sequences. Spatial coherence is captured via a Gaussian conditional random field in which edge weights are modulated by local differences in Lab color vectors, $\mathbf{I}^c_t(i,j)$:

$$E_{\text{space}}(\mathbf{g}_{tk}) = \frac{1}{2}\sum_{(i,j)}\sum_{(i',j')\in\Gamma(i,j)} w^{ij}_{i'j'}(g_{tk}(i,j) - g_{tk}(i',j'))^2, \tag{4}$$

$$w^{ij}_{i'j'} = \max\left\{\exp\left\{-\frac{1}{2\sigma_c^2}||\mathbf{I}^c_t(i,j) - \mathbf{I}^c_t(i',j')||^2\right\}, \delta_c\right\}. \tag{5}$$

The threshold $\delta_c > 0$ adds robustness to large color changes in internal object texture. Temporal coherence of surfaces is then encouraged via a corresponding Gaussian MRF:

$$E_{\text{time}}(\mathbf{g}_{tk}, \mathbf{g}_{t+1,k}, \mathbf{u}_{tk}, \mathbf{v}_{tk}) = \sum_{(i,j)}(g_{tk}(i,j) - g_{t+1,k}(i + u^{ij}_{tk}, j + v^{ij}_{tk}))^2. \tag{6}$$

Critically, this energy function uses the corresponding flow field to non-rigidly align the layers at subsequent frames. By allowing smooth deformation of the support functions $\mathbf{g}_{tk}$, we allow layer support to evolve over time, as opposed to transforming a single rigid template [12].

Our model of layer coherence is inspired by a recent method for image segmentation, based on spatially dependent Pitman-Yor processes [21]. That work makes connections between layered occlusion processes and *stick breaking* representations of nonparametric Bayesian models. By assigning appropriate stochastic priors to layer thresholds, the Pitman-Yor model captures the power law statistics of natural scene partitions and infers an appropriate number of segments for each image. Existing optical flow benchmarks employ artificially constructed scenes that may have different layer-level statistics. Consequently our experiments in this paper employ a fixed number of layers K.

3.3 Depth Ordering and Occlusion Reasoning

The preceding generative process defines a set of K ordered layers, with corresponding flow fields $\mathbf{u}_{tk}, \mathbf{v}_{tk}$ and segmentation masks $\mathbf{s}_{tk}$. Recall that the layer assignment masks $\mathbf{s}$ are a

deterministic function (threshold) of the underlying continuous layer support functions $\mathbf{g}$ (see Eq. (3)). To consistently reason about occlusions, we examine the layer assignments $s_{tk}(i,j)$ and $s_{t+1,k}(i+u_{tk}^{ij}, j+v_{tk}^{ij})$ at locations corresponded by the underlying flow fields. This leads to a far richer occlusion model than standard spatially independent outlier processes: geometric consistency is enforced via the layered sequence of flow fields.

Let $\mathbf{I}_t^s(i,j)$ denote an observed image feature for pixel (i,j); we work with a filtered version of the intensity images to provide some invariance to illumination changes. If $s_{tk}(i,j) = s_{t+1,k}(i+u_{tk}^{ij}, j+v_{tk}^{ij}) = 1$, the visible layer for pixel (i,j) at time t remains unoccluded at time $t+1$, and the image observations are modeled using a standard brightness (or, here, feature) constancy assumption. Otherwise, that pixel has become occluded, and is instead generated from a uniform distribution. The image likelihood model can then be written as

$$\begin{aligned} p(\mathbf{I}_t^s \mid \mathbf{I}_{t+1}^s, \mathbf{u}_t, \mathbf{v}_t, \mathbf{g}_t, \mathbf{g}_{t+1}) &\propto \exp\{-E_{\text{data}}(\mathbf{u}_t, \mathbf{v}_t, \mathbf{g}_t, \mathbf{g}_{t+1})\} \\ = \exp\Big\{ -\sum_k \sum_{(i,j)} \Big(&\rho_d(\mathbf{I}_t^s(i,j) - \mathbf{I}_{t+1}^s(i+u_{tk}^{ij}, j+v_{tk}^{ij})) s_{tk}(i,j) s_{t+1,k}(i+u_{tk}^{ij}, j+v_{tk}^{ij}) \\ &+ \lambda_d s_{tk}(i,j)(1 - s_{t+1,k}(i+u_{tk}^{ij}, j+v_{tk}^{ij})) \Big) \Big\} \end{aligned}$$

where $\rho_d(\cdot)$ is a robust potential function and the constant λ_d arises from the difference of the log normalization constants for the robust and uniform distributions. With algebraic simplifications, the data error term can be written as

$$\begin{aligned} &E_{\text{data}}(\mathbf{u}_t, \mathbf{v}_t, \mathbf{g}_t, \mathbf{g}_{t+1}) = \\ &\quad \sum_k \sum_{(i,j)} \Big(\rho_d(\mathbf{I}_t^s(i,j) - \mathbf{I}_{t+1}^s(i+u_{tk}^{ij}, j+v_{tk}^{ij})) - \lambda_d \Big) s_{tk}(i,j) s_{t+1,k}(i+u_{tk}^{ij}, j+v_{tk}^{ij}) \end{aligned} \quad (7)$$

up to an additive, constant multiple of λ_d. The shifted potential function $(\rho_d(\cdot) - \lambda_d)$ represents the change in energy when a pixel transitions from an occluded to an unoccluded configuration. Note that occlusions have higher likelihood only for sufficiently large discrepancies in matched image features and can only occur via a corresponding change in layer visibility.

4 Posterior Inference from Image Sequences

Considering the full generative model defined in Sec. 3, *maximum a posteriori* (MAP) estimation for a T frame image sequence is equivalent to minimization of the following energy function:

$$\begin{aligned} E(\mathbf{u}, \mathbf{v}, \mathbf{g}, \theta) = \sum_{t=1}^{T-1} \Big\{ &E_{\text{data}}(\mathbf{u}_t, \mathbf{v}_t, \mathbf{g}_t, \mathbf{g}_{t+1}) + \sum_{k=1}^{K} \lambda_a (E_{\text{aff}}(\mathbf{u}_{tk}, \theta_{tk}) + E_{\text{aff}}(\mathbf{v}_{tk}, \theta_{tk})) \\ &+ \sum_{k=1}^{K-1} \lambda_b E_{\text{space}}(\mathbf{g}_{tk}) + \lambda_c E_{\text{time}}(\mathbf{g}_{tk}, \mathbf{g}_{t+1,k}, \mathbf{u}_{tk}, \mathbf{v}_{tk}) \Big\} + \sum_{k=1}^{K-1} \lambda_b E_{\text{space}}(\mathbf{g}_{Tk}). \end{aligned} \quad (8)$$

Here λ_a, λ_b, and λ_c are weights controlling the relative importance of the affine, spatial, and temporal terms respectively. Simultaneously inferring flow fields, layer support maps, and depth ordering is a challenging process; our approach is summarized below.

4.1 Relaxation of the Layer Assignment Process

Due to the non-differentiability of the threshold process that determines assignments of regions to layers, direct minimization of Eq. (8) is challenging. For a related approach to image segmentation, a mean field variational method has been proposed [21]. However, that segmentation model is based on a much simpler, spatially factorized likelihood model for color and texture histogram features. Generalization to the richer flow likelihoods considered here raises significant complications.

Instead, we relax the hard threshold assignment process using the logistic function $\sigma(g) = 1/(1+\exp(-g))$. Applied to Eq. (3), this induces the following soft layer assignments:

$$\tilde{s}_{tk}(i,j) = \begin{cases} \sigma(\lambda_e g_{tk}(i,j)) \prod_{k'=1}^{k-1} \sigma(-\lambda_e g_{tk'}(i,j)), & 1 \le k < K, \\ \prod_{k'=1}^{K-1} \sigma(-\lambda_e g_{tk'}(i,j)), & k = K. \end{cases} \quad (9)$$

(a) (b) (c) (d) (e) (f) (g)

Figure 2: Results on the "Venus" sequence with 4 layers. The two background layers move faster than the two foreground layers, and the solution with the correct depth ordering has lower energy and smaller error. (a) First frame. (b-d) Fast-to-slow ordering: EPE 0.252 and energy -1.786×10^6. Left to right: motion segmentation, estimated flow field, and absolute error of estimated flow field. (f-g) Slow-to-fast ordering: EPE 0.195 and energy -1.808×10^6. Darker indicates larger flow field errors in (d) and (g).

Note that $\sigma(-g) = 1 - \sigma(g)$, and $\sum_{k=1}^{K} \tilde{s}_{tk}(i,j) = 1$ for any g_{tk} and constant $\lambda_e > 0$.

Substituting these soft assignments $\tilde{s}_{tk}(i,j)$ for $s_{tk}(i,j)$ in Eq. (7), we obtain a differentiable energy function that can be optimized via gradient-based methods. A related relaxation underlies the classic backpropagation algorithm for neural network training.

4.2 Gradient-Based Energy Minimization

We estimate the hidden fields for all the frames together, while fixing the flow fields, by optimizing an objective involving the relevant $E_{\text{data}}(\cdot)$, $E_{\text{space}}(\cdot)$, and $E_{\text{time}}(\cdot)$ terms. We then estimate the flow fields $\mathbf{u}_t, \mathbf{v}_t$ for each frame, while fixing those of neighboring frames and the hidden fields, via the $E_{\text{data}}(\cdot)$, $E_{\text{aff}}(\cdot)$, and $E_{\text{time}}(\cdot)$ terms. For flow estimation, we use a standard coarse-to-fine, warping-based technique as described in [22]. For hidden field estimation, we use an implementation of conjugate gradient descent with backtracking and line search. See *Supplemental Material* for details.

5 Experimental Results

We apply the proposed model to two-frame sequences and compute both the forward and backward flow fields. This enables the use of the temporal consistency term by treating one frame as both the previous and the next frame of the other[1]. We obtain an initial flow field using the Classic+NL method [22], cluster the flow vectors into K groups (layers), and convert the initial segmentation into the corresponding hidden fields. We then use a two-level Gaussian pyramid (downsampling factor 0.8) and perform a fairly standard incremental estimation of the flow fields for each layer. At each level, we perform 20 incremental warping steps and during each step alternately solve for the hidden fields and the flow estimates. In the end, we threshold the hidden fields to compute a hard segmentation, and obtain the final flow field by selecting the flow field from the appropriate layers.

Occluded regions are determined by inconsistencies between the hard segmentations at subsequent frames, as matched by the final flow field. We would ideally like to compare layer initializations based on all permutations of the initial flow vector clusters, but this would be computationally intensive for large K. Instead we compare two orders: a fast-to-slow order appropriate for rigid scenes, and an opposite slow-to-fast order (for variety and robustness). We illustrate automatic selection of the preferred order for the "Venus" sequence in Figure 2.

The parameters for all experiments are set to $\lambda_a = 3$, $\lambda_b = 30$, $\lambda_c = 4$, $\lambda_d = 9$, $\lambda_e = 2$, $\sigma_i = 12$, and $\delta_c = 0.004$. A generalized Charbonnier function is used for $\rho_S(\cdot)$ and $\rho_d(\cdot)$ (see *Supplemental Material*). Optimization takes about 5 hours for the two-frame "Urban" sequence using our MATLAB implementation.

5.1 Results on the Middlebury Benchmark

Training Set As a baseline, we implement the smoothness in layers model [26] using modern techniques, and obtain an average training end-point error (EPE) of 0.487. This is reasonable but not competitive with state-of-the-art methods. The proposed model with 1 to 4 layers produces average EPEs of 0.248, 0.212, 0.200, and 0.194, respectively (see Table 1). The one-layer model is similar to the Classic+NL method, but has a less sophisticated (more local) model of the flow within

[1]Our model works for longer sequences. We use two frames here for fair comparison with other methods.

Table 1: Average end-point error (EPE) on the Middlebury optical flow benchmark *training* set.

	Avg. EPE	Venus	Dimetrodon	Hydrangea	RubberWhale	Grove2	Grove3	Urban2	Urban3
Weiss [26]	0.487	0.510	0.179	0.249	0.236	0.221	0.608	0.614	1.276
Classic++	0.285	0.271	0.128	0.153	0.081	0.139	0.614	0.336	0.555
Classic+NL	0.221	0.238	0.131	0.152	0.073	0.103	0.468	0.220	0.384
1layer	0.248	0.243	0.144	0.175	0.095	0.125	0.504	0.279	0.422
2layers	0.212	0.219	0.147	0.169	0.081	0.098	0.376	0.236	0.370
3layers	0.200	0.212	0.149	0.173	0.073	0.090	0.343	0.220	0.338
4layers	0.194	0.197	0.148	0.159	0.068	0.088	0.359	0.230	0.300
3layers w/ WMF	0.195	0.211	0.150	0.161	0.067	0.086	0.331	0.210	0.345
3layers w/ WMF C++Init	0.203	0.212	0.151	0.161	0.066	0.087	0.339	0.210	0.396

Table 2: Average end-point error (EPE) on the Middlebury optical flow benchmark *test set*.

	Rank	**Average**	Army	Mequon	Schefflera	Wooden	Grove	Urban	Yosemite	Teddy
EPE										
Layers++	4.3	0.270	0.08	0.19	0.20	0.13	0.48	0.47	0.15	0.46
Classic+NL	6.5	0.319	0.08	0.22	0.29	0.15	0.64	0.52	0.16	0.49
EPE in boundary regions										
Layers++		0.560	0.21	0.56	0.40	0.58	0.70	1.01	0.14	0.88
Classic+NL		0.689	0.23	0.74	0.65	0.73	0.93	1.12	0.13	0.98

that layer. It thus performs worse than the Classic+NL initialization; the performance improvements allowed by additional layers demonstrate the benefits of a layered model.

Accuracy is improved by applying a 15×15 *weighted median filter* (WMF) [22] to the flow fields of each layer during the iterative warping step (EPE for 1 to 4 layers: 0.231, 0.204, 0.195, and 0.193). Weighted median filtering can be interpreted as a non-local spatial smoothness term in the energy function that integrates flow field information over a larger spatial neighborhood.

The "correct" number of layers for a real scene is not well defined (consider the "Grove3" sequence, for example). We use a restricted number of layers, and model the remaining complexity of the flow within each layer via the roughness-in-layers spatial term and the WMF. As the number of layers increases, the complexity of the flow within each layer decreases, and consequently the need for WMF also decreases; note that the difference in EPE for the 4-layer model with and without WMF is insignificant. For the remaining experiments we use the version with WMF.

To test the sensitivity of the result to the initialization, we also initialized with Classic++ ("C++Init" in Table 1), a good, but not top, non-layered method [22]. The average EPE for 1 to 4 layers increases to 0.248, 0.206, 0.203, and 0.198, respectively. While the one-layer method gets stuck in poor local minima on the "Grove3" and "Urban3" sequences, models with additional layers are more robust to the initialization. For more details and full EPE results, see the *Supplemental Material*.

Test Set For evaluation, we focus on a model with 3 layers (denoted "Layers++" in the Middlebury public table). On the Middlebury test set it has an average EPE of 0.270 and average angular error (AAE) of 2.556; this is the lowest among all tested methods [2] at the time of writing (Oct. 2010). Table 2 summarizes the results for individual test sequences. The layered model is particularly accurate at motion boundaries, probably due to the use of layer-specific motion models, and the explicit modeling of occlusion in E_{data} (Eq. (7)). For more extensive results, see the *Supplemental Material*.

Visual Comparison Figure 3 shows results for the 3-layer model on several training and test sequences. Notice that the layered model produces a motion segmentation that captures the major structure of the scene, and the layer boundaries correspond well to static image edges. It detects most occlusion regions and interpolates their motion reasonably well. Several sequences show significant improvement due to the global reasoning provided by the layered model. On the training "Grove3" sequence, the proposed method correctly identifies many holes between the branches and leaves as background. It also associates the branch at the bottom right corner with branches in the center. As the branch moves beyond the image boundary, the layered model interpolates its motion using long-range correlation with the branches in the center. In contrast, the single-layered approach incorrectly interpolates from local background regions. The "Schefflera" result illustrates how the layered method can separate foreground objects from the background (e.g., the leaves in the top right corner), and thereby reduce errors made by single-layer approaches such as Classic+NL.

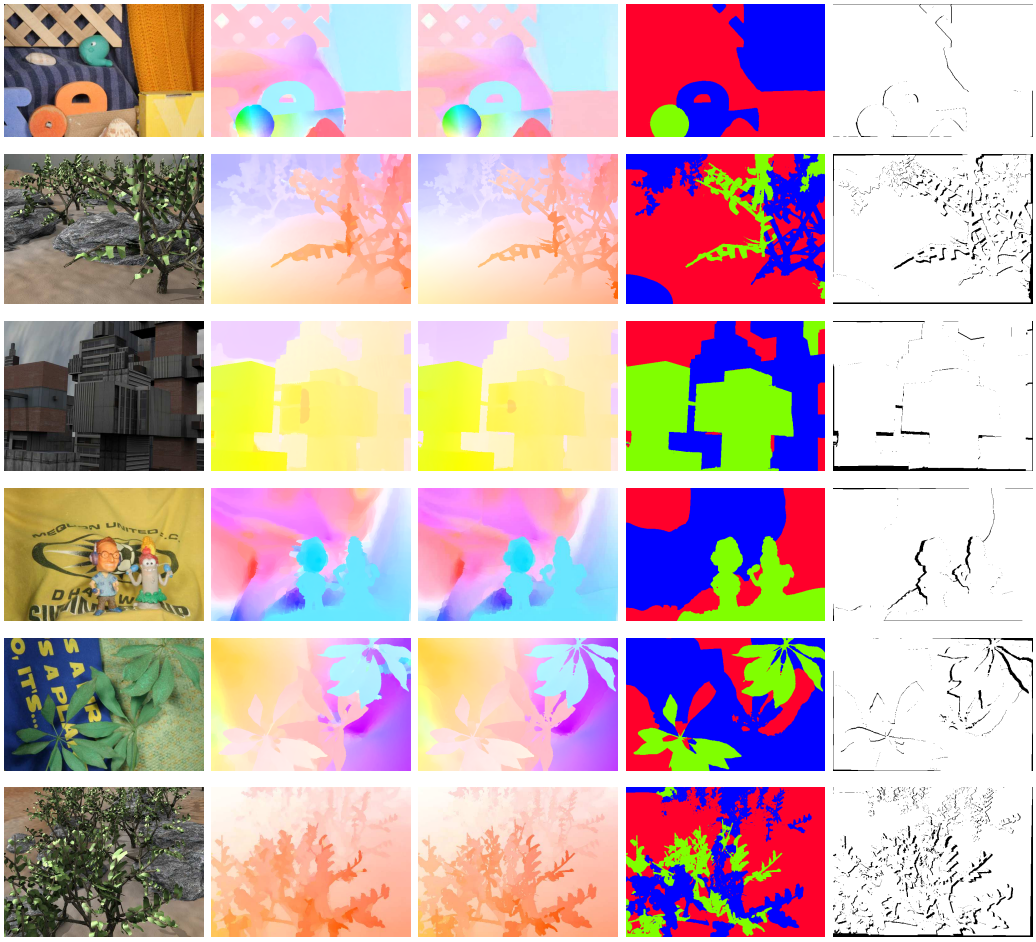


Figure 3: Results on some Middlebury training (rows 1 to 3) and test (rows 4 to 6) sequences. *Top to bottom:* "RubberWhale", "Grove3", "Urban3", "Mequon", "Schefflera", and "Grove". *Left to right:* First image frame, initial flow field from "Classic+NL", final flow field, motion segmentation (green front, blue middle, red back), and detected occlusions. Best viewed in color and enlarged to allow comparison of detailed motions.

6 Conclusion and Discussion

We have described a new probabilistic formulation for layered image motion that explicitly models occlusion and disocclusion, depth ordering of layers, and the temporal consistency of the layer segmentation. The approach allows the flow field in each layer to have piecewise smooth deformation from a parametric motion model. Layer support is modeled using an image-dependent hidden field prior that supports a model of temporal layer continuity over time. The image data error term takes into account layer occlusion relationships, resulting in increased flow accuracy near motion boundaries. Our method achieves state-of-the-art results on the Middlebury optical flow benchmark while producing meaningful segmentation and occlusion detection results.

Future work will address better inference methods, especially a better scheme to infer the layer order, and the automatic estimation of the number of layers. Computational efficiency has not been addressed, but will be important for inference on long sequences. Currently our method does not capture transparency, but this could be supported using a soft layer assignment and a different generative model. Additionally, the parameters of the model could be learned [23], but this may require more extensive and representative training sets. Finally, the parameters of the model, especially the number of layers, should adapt to the motions in a given sequence.

Acknowledgments DS and MJB were supported in part by the NSF Collaborative Research in Computational Neuroscience Program (IIS–0904875) and a gift from Intel Corp.

References

[1] S. Ayer and H. S. Sawhney. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and MDL encoding. In *ICCV*, pages 777–784, Jun 1995.

[2] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, to appear.

[3] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *ICCV*, pages 489–495, 1999.

[4] M. J. Black and P. Anandan. Robust dynamic motion estimation over time. In *CVPR*, pages 296–302, 1991.

[5] M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *CVIU*, 63:75–104, 1996.

[6] M. J. Black and A. D. Jepson. Estimating optical-flow in segmented images using variable-order parametric models with local deformations. *PAMI*, 18(10):972–986, October 1996.

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

[8] T. Darrell and A. Pentland. Cooperative robust estimation using layers of support. *PAMI*, 17(5):474–487, 1995.

[9] B. Glocker, T. H. Heibel, N. Navab, P. Kohli, and C. Rother. Triangleflow: Optical flow with triangulation-based higher-order likelihoods. In *ECCV*, pages 272–285, 2010.

[10] M. Irani, P. Anandan, and D. Weinshall. From reference frames to reference planes: Multi-view parallax geometry and applications. In *ECCV*, 1998.

[11] A. Jepson and M. J. Black. Mixture models for optical flow computation. In *CVPR*, 1993.

[12] N. Jojic and B. Frey. Learning flexible sprites in video layers. In *CVPR*, pages I:199–206, 2001.

[13] A. Kannan, B. Frey, and N. Jojic. A generative model of dense optical flow in layers. Technical Report TR PSI-2001-11, University of Toronto, Aug. 2001.

[14] R. Kumar, P. Anandan, and K. Hanna. Shape recovery from multiple views: A parallax based approach. In *Proc 12th ICPR*, 1994.

[15] R. D. Morris, X. Descombes, and J. Zerubia. The Ising/Potts model is not well suited to segmentation tasks. In *Proceedings of the IEEE Digital Signal Processing Workshop*, 1996.

[16] M. Nicolescu and G. Medioni. Motion segmentation with accurate boundaries - a tensor voting approach. In *CVPR*, pages 382–389, 2003.

[17] M. P. Kumar, P. H. Torr, and A. Zisserman. Learning layered motion segmentations of video. *IJCV*, 76(3):301–319, 2008.

[18] S. Roth and M. J. Black. On the spatial statistics of optical flow. *IJCV*, 74(1):33–50, August 2007.

[19] H. S. Sawhney. 3D geometry from planar parallax. In *CVPR*, pages 929–934, 1994.

[20] T. Schoenemann and D. Cremers. High resolution motion layer decomposition using dual-space graph cuts. In *CVPR*, pages 1–7, June 2008.

[21] E. Sudderth and M. Jordan. Shared segmentation of natural scenes using dependent Pitman-Yor processes. In *NIPS*, pages 1585–1592, 2009.

[22] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *CVPR*, 2010.

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

[24] P. Torr, R. Szeliski, and P. Anandan. An integrated Bayesian approach to layer extraction from image sequences. *PAMI*, 23(3):297–303, Mar 2001.

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

[26] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *CVPR*, pages 520–526, Jun 1997.

[27] Y. Weiss and E. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *CVPR*, pages 321–326, Jun 1996.

[28] M. Werlberger, T. Pock, and H. Bischof. Motion estimation with non-local total variation regularization. In *CVPR*, 2010.

[29] H. Yalcin, M. J. Black, and R. Fablet. The dense estimation of motion and appearance in layers. In *IEEE Workshop on Image and Video Registration*, pages 777–784, Jun 2004.

[30] Y. Zhou and H. Tao. Background layer model for object tracking through occlusion. In *ICCV*, volume 2, pages 1079–1085, 2003.

Layered Image Motion with Explicit Occlusions, Temporal Consistency, and Depth Ordering — Supplemental Material

Deqing Sun, Erik B. Sudderth, and Michael J. Black
Department of Computer Science, Brown University
{dqsun,sudderth,black}@cs.brown.edu

This document contains supplemental material for the paper

> Sun, D., Sudderth, E., and Black, M. J., "Layered image motion with explicit occlusions, temporal consistency, and depth ordering," *Advances in Neural Information Processing Systems 23*, NIPS, MIT Press, 2010.

Please see the main paper for a description of the optical flow model, its optimization, and the main results. Sections 1-3 below provide more details of the experiments in the main paper, as well as some additional results. Section 4 provides detailed formulas for the gradients of the energy function with respect to the optical flow and hidden layer support fields.

1 Implementation Details

To gain robustness against lighting changes, we follow [6] and apply the Rudin-Osher-Fatemi (ROF) structure texture decomposition method [3] to pre-process the input grayscale sequences. We linearly combine the texture and structure components in the proportion 20:1. The parameters are set according to [6].

We use the generalized Charbonnier penalty function $\rho(x) = (x^2 + \epsilon^2)^a$ with $\epsilon = 0.001$ and $a = 0.45$ [4] for $\rho_d(\cdot)$ in the data term, E_{data}, and $\rho_s(\cdot)$ in the spatial flow term, E_{aff}.

We compute the initial flow field using the Classic+NL method [4] and fit K affine motion fields to the initial forward flow field. The fitting method is similar to K-means, where we cluster the flow vectors and fit the affine parameters of each cluster. A pixel is visible at the layer that best explains its motion and invisible at the other layers. To avoid local minima, we perform 25 independent runs of the fitting method and select the result with the lowest fitting error. Warping the resultant segmentation using the backward flow field produces the initial segmentation of the next frame.

To convert the hard segmentation into the initial hidden fields, the kth ($k < K$) hidden field takes value 1.5 at pixels visible at the kth layer and -1.5 otherwise. Around occlusion/disocclusion regions, the layer assignment tends to change from one frame to the next. We detect pixels where the layer assignments, aligned by the initial flow field, disagree. For these pixels we divide their initial hidden field values by 10 to represent our uncertainty about the initial layer assignment in these occlusion/disocclusion regions. The initial motion of the visible pixels in each layer is the same as the initial flow field from Classic+NL, while the motion of the invisible pixels is interpolated by the fitted affine motion to the flow field.

2 More Experimental Results on the Middlebury Data Set

Table 1 provides the full end-point error (EPE) results for every training sequence using all the methods discussed in the main paper. We also evaluate several variants of the proposed method.

The results in the main paper are obtained with 20 warping steps per level, which is computationally expensive. Using 3 warping steps has slightly better overall performance for 1-3 layers, but 20 warping steps produces more accurate results in motion boundary regions.

To evaluate the method's sensitivity to the initialization, we compare the energy of the final solutions with initial flow fields from the Classic++ and the Classic+NL methods. As shown in Table 2, solutions with the Classic++ initialization have similar energy as those with the Classic+NL initialization. Table 1 shows that the average EPE obtained from both initializations is also similar. This suggests that the method works as long as the initialization is sensible. We expect that our current inference methods would not be able to recover from a really poor initialization.

Figure 3 and 4 show screen shots of the Middlebury public table for end-point error (EPE) and average angular error (AAE). The proposed method ("Layer++") is ranked first at the time of writing (end of Oct. 2010) and also has the lowest average EPE and average AAE both overall and in boundary regions.

We also provide the visual results of the proposed method on the training set in Figure 1 and on the test set in Figure 2. Layers++ reduces many of the errors made by the Classic+NL method in the motion boundary regions, produces sharp boundaries that correspond well to image structure, and is able to recover fine structures such as the leaf stems in the "Schefflera" sequence.

Finally average EPE and average AAE for all the test sequences are shown in Table 3.

Table 1: Average end-point error (EPE) on the Middlebury *training* set.

	Avg. EPE	Venus	Dimetrodon	Hydrangea	RubberWhale	Grove2	Grove3	Urban2	Urban3
Weiss [7]	0.487	0.510	0.179	0.249	0.236	0.221	0.608	0.614	1.276
Classic++	0.285	0.271	0.128	0.153	0.081	0.139	0.614	0.336	0.555
Classic+NL	0.221	0.238	0.131	0.152	0.073	0.103	0.468	0.220	0.384
20 warping steps (WS)									
1layer	0.248	0.243	0.144	0.175	0.095	0.125	0.504	0.279	0.422
2layers	0.212	0.219	0.147	0.169	0.081	0.098	0.376	0.236	0.370
3layers	0.200	0.212	0.149	0.173	0.073	0.090	0.343	0.220	0.338
4layers	0.194	0.197	0.148	0.159	0.068	0.088	0.359	0.230	0.300
5layers	0.196	0.195	0.151	0.169	0.063	0.086	0.345	0.211	0.351
20 WS w/ WMF : overall (method from main paper)									
1layer	0.231	0.235	0.144	0.155	0.075	0.106	0.462	0.245	0.426
2layers	0.204	0.217	0.149	0.156	0.070	0.090	0.357	0.219	0.373
3layers	0.195	0.211	0.150	0.161	0.067	0.086	0.331	0.210	0.345
4layers	0.193	0.195	0.150	0.155	0.064	0.087	0.351	0.222	0.321
5layers	0.197	0.196	0.149	0.173	0.065	0.087	0.347	0.214	0.346
20 WS w/ WMF: boundary region									
1layer	0.545	0.617	0.222	0.379	0.218	0.295	0.868	0.703	1.061
2layers	0.468	0.456	0.250	0.390	0.206	0.231	0.652	0.670	0.889
3layers	0.451	0.441	0.252	0.409	0.197	0.220	0.596	0.610	0.885
4layers	0.436	0.348	0.250	0.393	0.182	0.230	0.636	0.647	0.801
5layers	0.437	0.345	0.250	0.438	0.182	0.221	0.626	0.602	0.834
3 WS w/ WMF: overall									
1layer	0.219	0.231	0.119	0.152	0.074	0.097	0.454	0.230	0.394
2layers	0.195	0.211	0.122	0.159	0.070	0.084	0.364	0.205	0.346
3layers	0.190	0.212	0.128	0.163	0.066	0.080	0.347	0.206	0.321
4layers	0.194	0.192	0.132	0.158	0.063	0.081	0.365	0.227	0.337
5layers	0.196	0.192	0.136	0.159	0.063	0.080	0.362	0.224	0.349
3 WS w/ WMF: boundary region									
1layer	0.551	0.642	0.218	0.385	0.229	0.291	0.859	0.710	1.074
2layers	0.472	0.464	0.236	0.414	0.218	0.238	0.672	0.662	0.876
3layers	0.463	0.441	0.254	0.428	0.207	0.221	0.630	0.632	0.891
4layers	0.465	0.353	0.264	0.415	0.187	0.229	0.665	0.671	0.934
5layers	0.466	0.354	0.271	0.418	0.191	0.220	0.653	0.662	0.962
20 WS w/ WMF: Classic++ init									
1layer	0.248	0.232	0.144	0.155	0.079	0.107	0.523	0.261	0.487
2layers	0.206	0.218	0.149	0.156	0.072	0.090	0.373	0.218	0.372
3layers	0.203	0.212	0.151	0.161	0.066	0.087	0.339	0.210	0.396
4layers	0.198	0.195	0.149	0.155	0.064	0.087	0.342	0.229	0.360
5layers	0.192	0.194	0.148	0.161	0.063	0.085	0.326	0.231	0.327

Table 2: Energy ($\times 10^6$) of the solutions obtained by the proposed method with three layers. The energy is shown for all the sequences in the *training* set using two different initializations.

	Venus	Dimetrodon	Hydrangea	RubberWhale	Grove2	Grove3	Urban2	Urban3
"Classic+NL" Init	-1.814	-2.609	-2.370	-3.039	-2.679	-1.979	-3.198	-3.044
"Classic++" Init	-1.814	-2.613	-2.369	-3.039	-2.680	-1.974	-3.200	-2.998

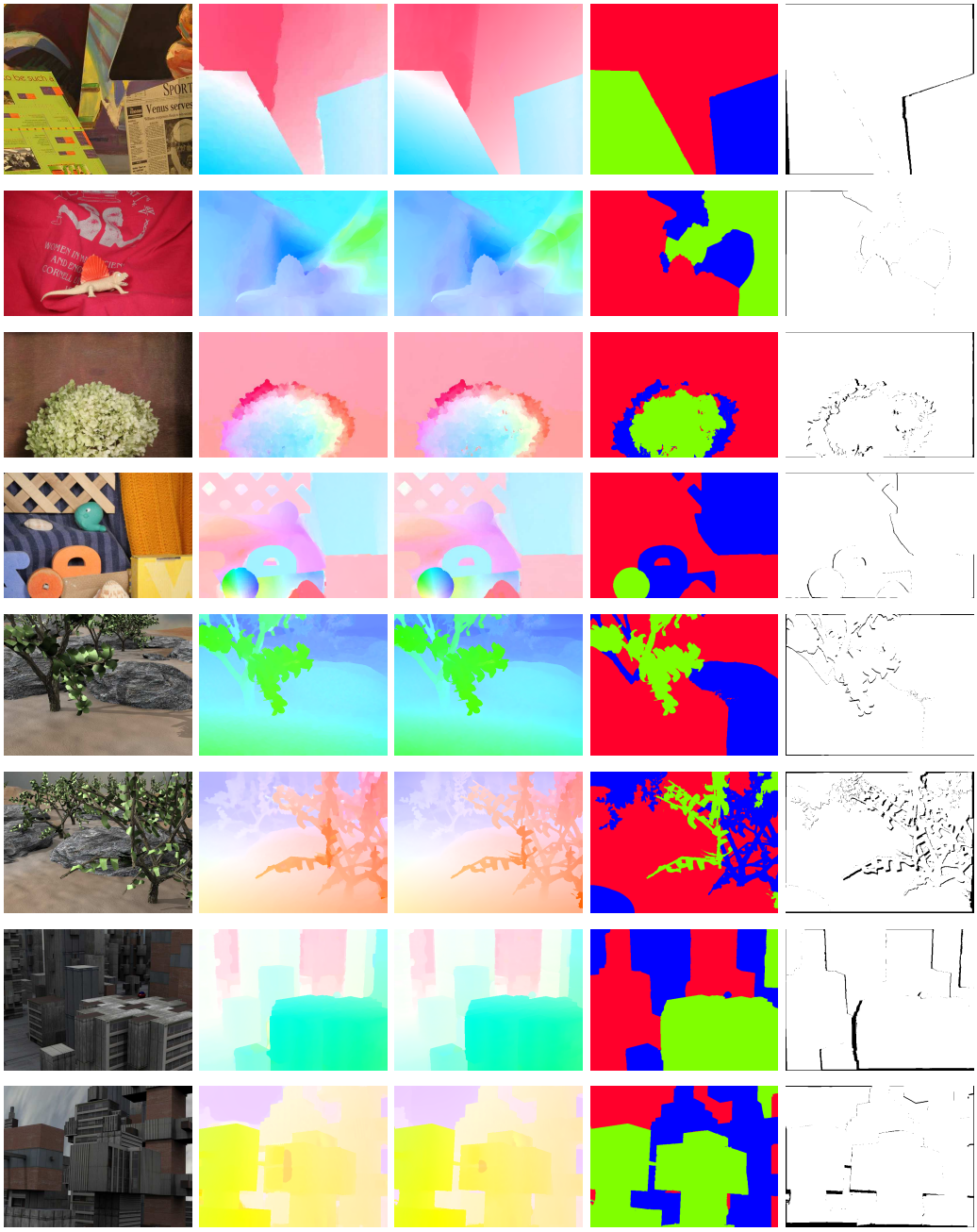


Figure 1: Results on the Middlebury *training* set. Left to right: first image, initial flow field given by Classic+NL, final flow field, motion segmentation, and detected occlusions (black). Best viewed in color and better enlarged for comparing the flow fields.

3 Results on the "Hand" Sequence

While the method performs well on the Middlebury evaluation, how well do the results generalize to other sequences? To find out, we apply the proposed model with 3 layers to the challenging "Hand" sequence [1], as shown in Figure 5. With the parameter settings tuned to the Middlebury training sequences, the proposed model does not recover the regions between fingers (Figure 5, top row). With a different parameter setting ($\lambda_d = 5$, and $\lambda_b = 90$), the proposed model can successfully recover the regions between fingers. The EPE for this sequence drops from 2.754 to

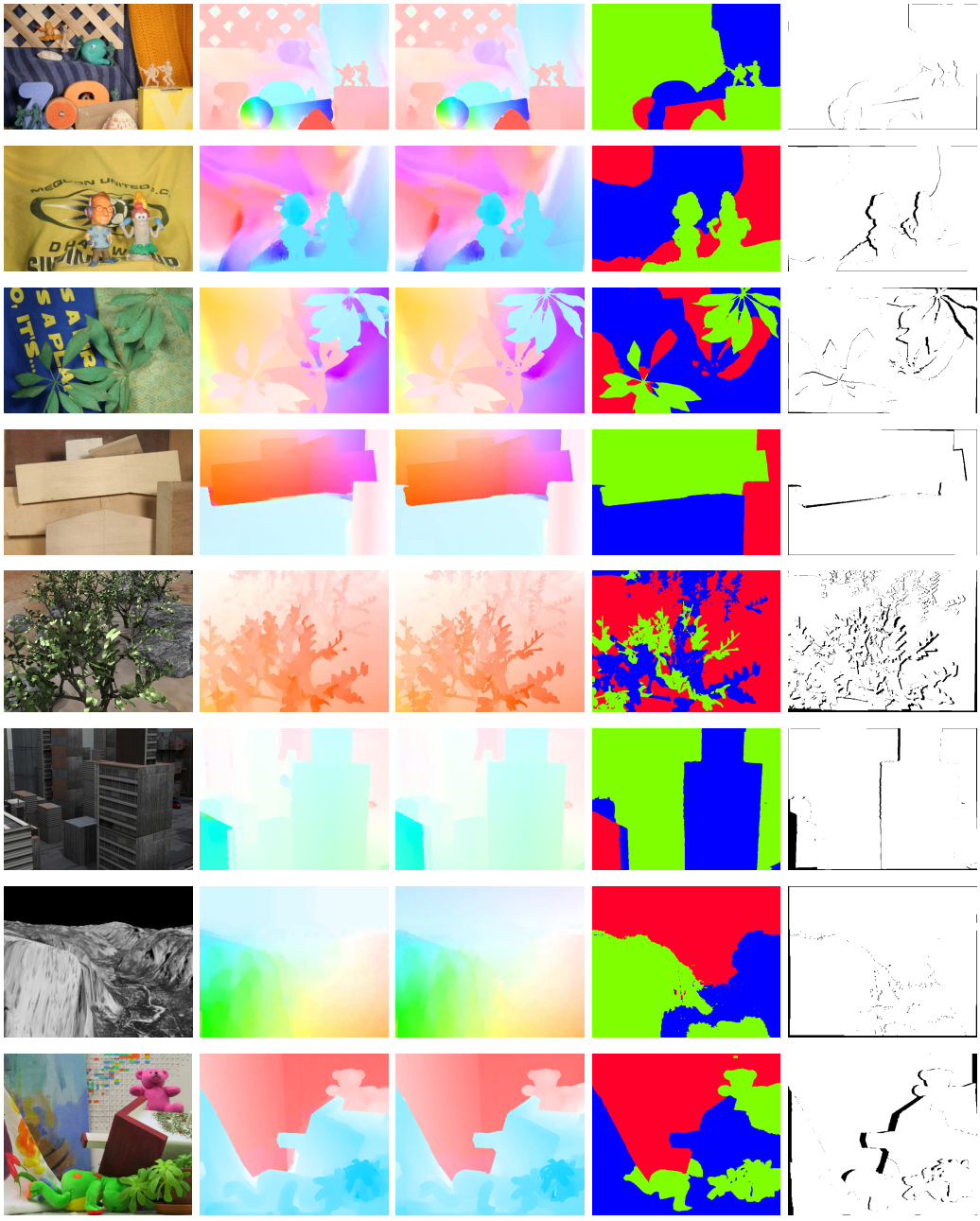
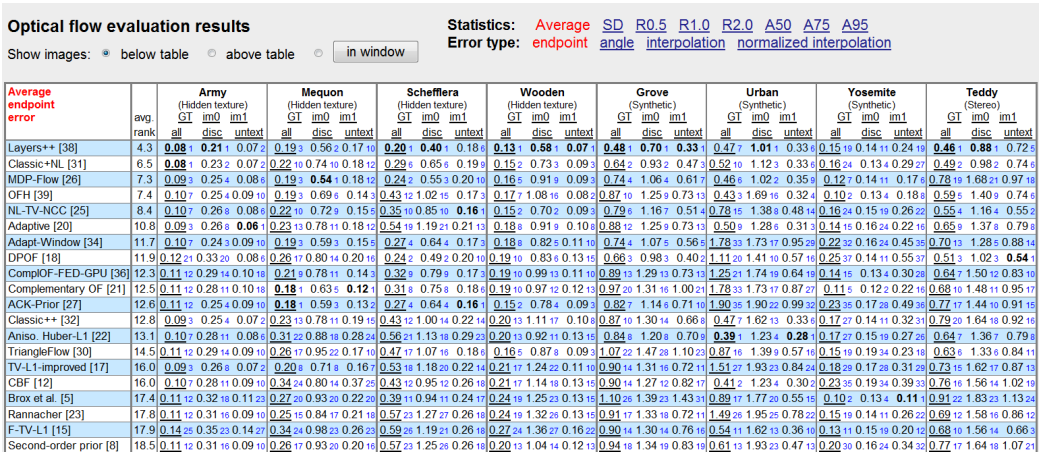


Figure 2: Results on the Middlebury *test* set. Left to right: first image, initial flow field given by Classic+NL, final flow field, motion segmentation, and detected occlusions (black). Best viewed in color and better enlarged for comparing the flow fields.

1.909. Moreover, note that the model successfully recovers from failures of the initialization in the regions between the fingers.

Table 4 compares this new parameter settings with the old settings on the Middlebury training sequences. The new settings produces an average training EPE of 0.215 which is about 10% worse than the result reported in the main paper.

This suggests that the proposed method may suffer from over fitting to the Middlebury evaluation. Future work should consider learning the parameters using a more representative data set [5] and automatically adapting the parameters to a particular sequence.



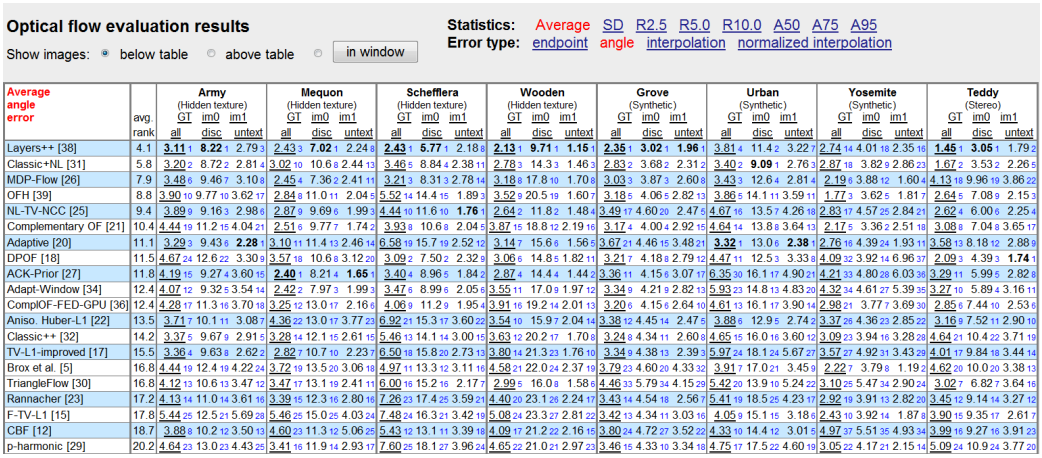
Optical flow evaluation results

Show images: below table above table in window

Statistics: Average SD R0.5 R1.0 R2.0 A50 A75 A95
Error type: endpoint angle interpolation normalized interpolation

Average endpoint error	avg. rank	Army (Hidden texture) GT im0 im1			Mequon (Hidden texture) GT im0 im1			Schefflera (Hidden texture) GT im0 im1			Wooden (Hidden texture) GT im0 im1			Grove (Synthetic) GT im0 im1			Urban (Synthetic) GT im0 im1			Yosemite (Synthetic) GT im0 im1			Teddy (Stereo) GT im0 im1		
		all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext
Layers++ [38]	4.3	**0.08** 1	**0.21** 1	0.07 2	0.19 3	0.56 2	0.17 10	**0.20** 1	**0.40** 1	0.18 6	**0.13** 1	**0.58** 1	**0.07** 1	**0.48** 1	**0.70** 1	**0.33** 1	0.47 7	**1.01** 1	0.33 6	0.15 19	0.14 11	0.24 19	**0.46** 1	**0.88** 1	0.72 5
Classic+NL [31]	6.5	**0.08** 1	0.23 2	0.07 2	0.22 10	0.74 10	0.18 12	0.29 6	0.65 6	0.19 9	0.15 2	0.73 3	0.09 3	0.64 2	0.93 2	0.47 3	0.52 10	1.12 3	0.33 6	0.16 24	0.13 4	0.29 27	0.49 2	0.98 2	0.74 6
MDP-Flow [26]	7.3	0.09 3	0.25 4	0.08 6	0.19 3	**0.54** 1	0.18 12	0.24 2	0.55 3	0.20 10	0.16 5	0.91 9	0.09 3	0.74 4	1.06 4	0.61 7	0.46 6	1.02 2	0.35 8	0.12 7	0.14 11	0.17 8	0.78 19	1.68 21	0.97 18
OFH [39]	7.4	0.10 7	0.25 4	0.09 10	0.19 3	0.69 8	0.14 3	0.43 12	1.02 15	0.17 3	0.17 7	1.08 16	0.08 2	0.87 10	1.25 9	0.73 13	0.43 3	1.69 16	0.32 4	0.10 2	0.13 4	0.18 8	0.59 5	1.40 9	0.74 6
NL-TV-NCC [25]	8.4	0.10 7	0.26 8	0.08 6	0.22 10	0.72 9	0.15 5	0.35 10	0.85 10	**0.16** 1	0.15 2	0.70 2	0.09 3	0.79 8	1.16 7	0.51 4	0.78 15	1.38 8	0.48 14	0.16 24	0.15 19	0.26 22	0.55 4	1.16 4	0.55 2
Adaptive [20]	10.8	0.09 3	0.26 8	**0.06** 1	0.23 13	0.78 11	0.18 12	0.54 19	1.19 21	0.21 13	0.18 8	0.91 9	0.10 8	0.88 12	1.25 9	0.73 13	0.50 9	1.28 6	0.31 3	0.14 15	0.16 24	0.22 16	0.65 9	1.37 8	0.79 8
Adapt-Window [34]	11.7	0.10 7	0.24 3	0.09 10	0.19 3	0.59 3	0.15 5	0.27 4	0.64 4	0.17 3	0.18 8	0.82 5	0.11 10	0.74 4	1.07 5	0.56 6	1.78 33	1.73 17	0.95 29	0.22 32	0.16 24	0.45 35	0.70 13	1.28 6	0.88 14
DPOF [18]	11.9	0.12 21	0.33 20	0.08 6	0.26 17	0.80 14	0.20 16	0.24 2	0.49 2	0.20 10	0.19 10	0.83 6	0.13 15	0.66 3	0.98 3	0.40 2	1.11 20	1.41 10	0.57 16	0.25 37	0.14 11	0.55 37	0.51 3	1.02 3	**0.54** 1
ComplOF-FED-GPU [36]	12.3	0.11 12	0.29 14	0.10 18	0.21 9	0.78 11	0.14 3	0.32 9	0.79 9	0.17 3	0.19 10	0.99 13	0.11 10	0.89 13	1.29 13	0.73 13	1.25 21	1.74 19	0.64 19	0.14 15	0.13 4	0.30 28	0.64 7	1.50 12	0.83 10
Complementary OF [21]	12.5	0.11 12	0.28 11	0.10 18	**0.18** 1	0.63 5	**0.12** 1	0.31 8	0.75 8	0.18 6	0.19 10	0.97 12	0.12 13	0.97 20	1.31 16	1.00 21	1.78 33	1.73 17	0.87 27	0.11 5	0.12 2	0.22 16	0.68 10	1.48 11	0.95 17
ACK-Prior [27]	12.6	0.11 12	0.25 4	0.09 10	**0.18** 1	0.59 3	0.13 2	0.27 4	0.64 4	**0.16** 1	0.15 2	0.78 4	0.09 3	0.82 7	1.14 6	0.71 10	1.90 35	1.90 22	0.99 32	0.23 35	0.17 28	0.49 36	0.77 17	1.44 10	0.91 15
Classic++ [32]	12.8	0.09 3	0.25 4	0.07 2	0.23 13	0.78 11	0.19 15	0.43 12	1.00 14	0.22 14	0.20 13	1.11 17	0.10 8	0.87 10	1.30 14	0.66 8	0.47 7	1.62 13	0.33 6	0.17 27	0.14 11	0.32 31	0.79 20	1.64 18	0.92 16
Aniso. Huber-L1 [22]	13.1	0.10 7	0.28 11	0.08 6	0.31 22	0.88 18	0.28 24	0.56 21	1.13 18	0.29 23	0.20 13	0.92 11	0.13 15	0.84 8	1.20 8	0.70 9	**0.39** 1	1.23 4	**0.28** 1	0.17 27	0.15 19	0.27 26	0.64 7	1.36 7	0.79 8
TriangleFlow [30]	14.5	0.11 12	0.29 14	0.09 10	0.26 17	0.95 22	0.17 10	0.47 17	1.07 16	0.18 6	0.16 5	0.87 8	0.09 3	1.07 22	1.47 28	1.10 23	0.87 16	1.39 9	0.57 16	0.15 19	0.19 34	0.23 18	0.63 6	1.33 6	0.84 11
TV-L1-improved [17]	16.0	0.09 3	0.26 8	0.07 2	0.20 8	0.71 8	0.16 7	0.53 18	1.18 20	0.22 14	0.21 17	1.24 22	0.11 10	0.90 14	1.31 16	0.72 11	1.51 27	1.93 23	0.84 24	0.18 29	0.17 28	0.31 29	0.73 15	1.62 17	0.87 13
CBF [12]	16.0	0.10 7	0.28 11	0.09 10	0.34 24	0.80 14	0.37 25	0.43 12	0.95 12	0.26 18	0.21 17	1.14 18	0.13 15	0.90 14	1.27 12	0.82 17	0.41 2	1.23 4	0.30 2	0.23 35	0.19 34	0.39 33	0.76 16	1.56 14	1.02 19
Brox et al. [5]	17.4	0.11 12	0.32 18	0.11 23	0.27 20	0.93 20	0.22 20	0.39 11	0.94 11	0.24 17	0.24 19	1.25 23	0.13 15	1.10 26	1.39 23	1.43 31	0.89 17	1.77 20	0.55 15	0.10 2	0.13 4	**0.11** 1	0.91 22	1.83 23	1.13 24
Rannacher [23]	17.8	0.11 12	0.31 16	0.09 10	0.25 15	0.84 17	0.21 18	0.57 23	1.27 27	0.26 18	0.24 19	1.32 26	0.13 15	0.91 17	1.33 18	0.72 11	1.49 26	1.95 25	0.78 22	0.15 19	0.14 11	0.26 22	0.69 12	1.58 16	0.86 12
F-TV-L1 [15]	17.9	0.14 25	0.35 23	0.14 27	0.34 24	0.98 23	0.26 23	0.59 26	1.19 21	0.26 18	0.27 24	1.36 27	0.16 22	0.90 14	1.30 14	0.76 16	0.54 11	1.62 13	0.36 10	0.13 11	0.15 19	0.20 12	0.68 10	1.56 14	0.66 3
Second-order prior [8]	18.5	0.11 12	0.31 16	0.09 10	0.26 17	0.93 20	0.20 16	0.57 23	1.25 26	0.26 18	0.20 13	1.04 14	0.12 13	0.94 18	1.34 19	0.83 19	0.61 13	1.93 23	0.47 13	0.20 30	0.16 24	0.34 32	0.77 17	1.64 18	1.07 21

Figure 3: Screen shot of the Middlebury public end-point error (EPE) table; the proposed method is "Layer++".



Optical flow evaluation results

Show images: below table above table in window

Statistics: Average SD R2.5 R5.0 R10.0 A50 A75 A95
Error type: endpoint angle interpolation normalized interpolation

Average angle error	avg. rank	Army (Hidden texture) GT im0 im1			Mequon (Hidden texture) GT im0 im1			Schefflera (Hidden texture) GT im0 im1			Wooden (Hidden texture) GT im0 im1			Grove (Synthetic) GT im0 im1			Urban (Synthetic) GT im0 im1			Yosemite (Synthetic) GT im0 im1			Teddy (Stereo) GT im0 im1		
		all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext
Layers++ [38]	4.1	**3.11** 1	**8.22** 1	2.79 3	2.43 3	**7.02** 1	2.24 8	**2.43** 1	**5.77** 1	2.18 8	**2.13** 1	**9.71** 1	**1.15** 1	**2.35** 1	**3.02** 1	**1.96** 1	3.81 4	11.4 2	3.22 7	2.74 14	4.01 18	2.35 16	**1.45** 1	**3.05** 1	1.79 2
Classic+NL [31]	5.8	3.20 2	8.72 2	2.81 4	3.02 10	10.6 8	2.44 13	3.46 5	8.84 4	2.38 11	2.78 3	14.3 3	1.46 3	2.83 2	3.68 2	2.31 2	3.40 2	**9.09** 1	2.76 3	2.87 18	3.82 9	2.86 23	1.67 2	3.53 2	2.26 5
MDP-Flow [26]	7.9	3.48 6	9.46 7	3.10 8	2.45 4	7.36 2	2.41 11	3.21 3	8.31 3	2.78 14	3.18 8	17.8 10	1.70 8	3.03 3	3.87 3	2.60 8	3.43 3	12.6 4	2.81 4	2.19 8	3.88 12	1.60 4	4.13 18	9.96 19	3.86 22
OFH [39]	8.8	3.90 10	9.77 10	3.62 17	2.84 8	11.0 11	2.04 5	5.52 14	14.4 15	1.89 3	3.52 9	20.5 19	1.60 7	3.18 5	4.06 5	2.82 13	3.86 5	14.1 11	3.59 11	1.77 3	3.62 5	1.81 7	2.64 5	7.08 9	2.15 3
NL-TV-NCC [25]	9.4	3.89 9	9.16 3	2.98 6	2.87 9	9.69 6	1.99 3	4.44 10	11.6 10	**1.76** 1	2.64 2	11.8 2	1.48 4	3.49 17	4.60 20	2.47 5	4.67 16	13.5 7	4.26 18	2.83 17	4.57 26	2.84 21	2.62 4	6.00 6	2.25 4
Complementary OF [21]	10.4	4.44 19	11.2 15	4.04 21	2.51 6	9.77 7	1.74 2	3.93 8	10.6 8	2.04 5	3.87 15	18.8 12	2.19 16	3.17 4	4.00 4	2.92 15	4.64 14	13.8 8	3.64 13	2.17 5	3.36 2	2.51 18	3.08 8	7.04 8	3.65 17
Adaptive [20]	11.1	3.29 3	9.43 6	**2.28** 1	3.10 11	11.4 13	2.46 14	6.58 19	15.7 19	2.52 12	3.14 7	15.6 6	1.56 5	3.67 21	4.46 15	3.48 21	**3.32** 1	13.0 6	**2.38** 1	2.76 16	4.39 24	1.93 11	3.58 13	8.18 12	2.88 9
DPOF [18]	11.5	4.67 24	12.6 22	3.30 9	3.57 18	10.6 8	3.12 20	3.09 2	7.50 2	2.32 9	3.06 6	14.8 5	1.82 11	3.21 7	4.18 8	2.79 12	4.47 11	12.5 3	3.33 8	4.09 32	3.92 14	6.96 37	2.09 3	4.39 3	**1.74** 1
ACK-Prior [27]	11.8	4.19 15	9.27 4	3.60 15	**2.40** 1	8.21 4	**1.65** 1	3.40 4	8.96 5	1.84 2	2.87 4	14.4 4	1.44 2	3.36 11	4.15 6	3.07 17	6.35 30	16.1 17	4.90 21	4.21 33	4.80 28	6.03 36	3.29 11	5.99 5	2.82 8
Adapt-Window [34]	12.4	4.07 12	9.32 5	3.54 14	2.42 2	7.97 3	1.99 3	3.47 6	8.99 6	2.05 6	3.55 11	17.0 9	1.97 12	3.34 9	4.21 9	2.82 13	5.93 23	14.8 13	4.83 20	4.32 34	4.61 27	5.39 35	3.27 10	5.89 4	3.16 11
ComplOF-FED-GPU [36]	12.4	4.28 17	11.3 16	3.70 18	3.25 12	13.0 17	2.16 6	4.06 9	11.2 9	1.95 4	3.91 16	19.2 14	2.01 13	3.20 6	4.15 6	2.64 10	4.61 13	16.1 17	3.90 14	2.98 21	3.77 7	3.69 30	2.85 6	7.44 10	2.53 6
Aniso. Huber-L1 [22]	13.5	3.71 7	10.1 11	3.08 7	4.36 22	13.0 17	3.77 23	6.92 21	15.3 17	3.60 22	3.54 10	15.9 7	2.04 14	3.38 12	4.45 14	2.47 5	3.88 8	12.9 5	2.74 2	3.37 26	4.36 23	2.85 22	3.16 9	7.52 11	2.90 10
Classic++ [32]	14.2	3.37 5	9.67 9	2.91 5	3.28 14	12.1 15	2.61 15	5.46 13	14.1 14	3.00 15	3.63 12	20.2 17	1.70 8	3.24 8	4.34 11	2.60 8	4.65 15	16.0 16	3.60 12	3.09 23	3.94 16	3.28 28	4.64 21	10.4 22	3.71 19
TV-L1-improved [17]	15.5	3.36 4	9.63 8	2.62 2	2.82 7	10.7 10	2.23 7	6.50 18	15.8 20	2.73 13	3.80 14	21.3 23	1.76 10	3.34 9	4.38 13	2.39 3	5.97 24	18.1 24	5.67 27	3.57 27	4.92 31	3.43 29	4.01 17	9.84 18	3.44 14
Brox et al. [5]	16.8	4.44 19	12.4 19	4.22 24	3.72 19	13.5 20	3.06 18	4.97 11	13.3 12	3.11 16	4.58 21	22.0 24	2.37 19	3.79 23	4.60 20	4.33 32	3.91 7	17.0 21	3.45 9	2.22 7	3.79 8	1.19 2	4.62 20	10.0 20	3.38 13
TriangleFlow [30]	16.8	4.12 13	10.6 13	3.47 12	3.47 17	13.1 19	2.41 11	6.00 16	15.2 16	2.17 7	2.99 5	16.0 8	1.58 6	4.46 33	5.79 34	4.15 29	5.42 20	13.9 10	5.24 22	3.10 25	5.47 34	2.90 24	3.02 7	6.82 7	3.64 16
Rannacher [23]	17.2	4.13 14	11.0 14	3.61 16	3.39 15	12.3 16	2.80 16	7.26 23	17.4 25	3.59 21	4.40 20	23.1 26	2.24 17	3.43 14	4.54 18	2.56 7	5.41 19	18.5 25	4.23 17	2.92 19	3.91 13	2.82 20	3.45 12	9.14 14	3.27 12
F-TV-L1 [15]	17.8	5.44 25	12.5 21	5.69 28	5.46 25	15.0 25	4.03 24	7.48 24	16.3 21	3.42 19	5.08 24	23.3 27	2.81 22	3.42 13	4.34 11	3.03 16	4.05 9	15.1 15	3.18 6	2.43 10	3.92 14	1.87 8	3.90 15	9.35 17	2.61 7
CBF [12]	18.7	3.88 8	10.2 12	3.50 13	4.60 23	11.3 12	5.06 25	5.43 12	13.1 11	3.39 18	4.09 17	21.2 22	2.16 15	3.80 24	4.72 27	3.52 22	4.33 10	14.4 12	3.01 5	4.97 37	5.51 35	4.93 34	3.99 18	9.27 16	3.91 23
p-harmonic [29]	20.2	4.64 23	13.0 23	4.43 25	3.41 16	11.9 14	2.93 17	7.60 25	18.1 27	3.96 24	4.65 22	21.0 21	2.97 23	3.46 15	4.33 10	3.34 18	4.75 17	17.5 22	4.60 19	3.05 22	4.17 21	2.15 14	5.09 24	10.9 24	3.77 20

Figure 4: Screen shot of the Middlebury public average angular error (AAE) table; the proposed method is "Layer++".

4 Gradients of the Energy Function w. r. t. the Flow and the Hidden Fields

We use gradient-based methods to optimize the proposed energy function and this section summarizes the gradients with respect to the flow and the hidden fields. We derive the gradient for each

Table 3: Average end-point error (EPE) and angular error (AAE) on the Middlebury optical flow benchmark (*test set*).

	Rank	**Average**	Army	Mequon	Schefflera	Wooden	Grove	Urban	Yosemite	Teddy
EPE										
Layers++	4.3	0.270	0.08	0.19	0.20	0.13	0.48	0.47	0.15	0.46
Classic+NL	6.5	0.319	0.08	0.22	0.29	0.15	0.64	0.52	0.16	0.49
EPE in boundary regions										
Layers++		0.560	0.21	0.56	0.40	0.58	0.70	1.01	0.14	0.88
Classic+NL		0.689	0.23	0.74	0.65	0.73	0.93	1.12	0.13	0.98
AAE										
Layers++	4.1	2.556	3.11	2.43	2.43	2.13	2.35	3.81	2.74	1.45
Classic+NL	5.8	2.904	3.20	3.02	3.46	2.78	2.83	3.40	2.87	1.67
AAE in boundary regions										
Layers++		6.525	8.22	7.02	5.77	9.71	3.02	11.40	4.01	3.05
Classic+NL		7.823	8.72	10.60	8.84	14.30	3.68	9.09	3.82	3.53

Table 4: Average end-point error (EPE) on the Middlebury *training* set by the proposed model with 3 layers and two different sets of parameters.

	Avg. EPE	Venus	Dimetrodon	Hydrangea	RubberWhale	Grove2	Grove3	Urban2	Urban3
3layers ($\lambda_b = 10, \lambda_d = 9$)	0.195	0.211	0.150	0.161	0.067	0.086	0.331	0.210	0.345
3layers ($\lambda_b = 90, \lambda_d = 5$)	0.215	0.210	0.155	0.169	0.071	0.090	0.373	0.273	0.379

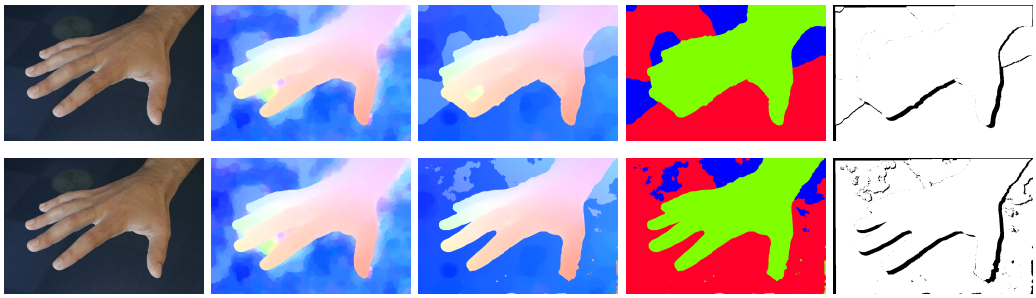


Figure 5: Results on the "Hand" sequence. Top: using same parameters as those used for the Middlebury data set (EPE 2.754). Bottom: using parameters tuned for hand (EPE 1.909). Left to right: first image, initial flow field given by Classic+NL, final flow field, motion segmentation, and detected occlusions (black). Best viewed in color.

individual term in the objective. From these it is easy to obtain the formula for the overall objective. Most of the derivations are straightforward and we only elaborate where there are subtle points.

4.1 Gradients w. r. t. the Hidden Field

Temporal Coherence Term The hidden field $\mathbf{g}_{t,k}$ appears in the temporal coherence term at time t and $t-1$. For the one at time t,

$$\frac{\partial E_{\text{time}}(\mathbf{g}_{t,k}, \mathbf{g}_{t+1,k}, \mathbf{u}_{tk}, \mathbf{v}_{tk})}{\partial \mathbf{g}_{tk}(i,j)} = 2(g_{tk}(i,j) - g_{t+1,k}(i + u_{tk}^{ij}, j + v_{tk}^{ij})). \tag{1}$$

The term $E_{\text{time}}(\mathbf{g}_{t-1,k}, \mathbf{g}_{t,k}, \mathbf{u}_{t-1,k}, \mathbf{v}_{t-1,k})$ involves the expression $g_{tk}(i + u_{t-1,k}^{ij}, j + v_{t-1,k}^{ij})$, which we compute using bi-linear interpolation. This is a linear operation applied to the hidden field $\mathbf{g}_{tk}$ and we can express it as a matrix, $\mathbf{W}_{t-1,k}$, applied to the vectorized hidden field $\mathbf{g}_{t,k}(:)$. Here ":" means arranging the matrix into a vector in a column-major way; that is, in the same way as the MATLAB vectorization operator. Now $\mathbf{W}_{t-1,k}\mathbf{g}_{t,k}(:)$ is the vectorized, warped result using the flow field from frame $t-1$ to frame t and

$$E_{\text{time}}(\mathbf{g}_{t-1,k}, \mathbf{g}_{tk}) = ||\mathbf{g}_{t-1,k}(:) - \mathbf{W}_{t-1,k}\mathbf{g}_{tk}(:)||^2. \tag{2}$$

Its gradient w. r. t. $\mathbf{g}_{tk}(:)$ is

$$\nabla_{\mathbf{g}_{tk}} E_{\text{time}}(\mathbf{g}_{t-1,k}, \mathbf{g}_{tk}) = 2\mathbf{W}_{t-1,k}^T(\mathbf{W}_{t-1,k}\mathbf{g}_{tk}(:) - \mathbf{g}_{t-1,k}(:)). \tag{3}$$

Note that there is no need to construct the matrix $\mathbf{W}_{t-1,k}$ in the implementation and we just need to perform the interpolation operation and its transpose.

Data Term Similarly, the hidden field $\mathbf{g}_{t,k}$ appears in the data term at time t and $t-1$. What is subtle is that the hidden field of a front layer influences the data term of the layers behind. We will first give the gradient of the soft layer assignment w. r. t. the hidden field, which plays a major role of the later derivations. Recall that the soft layer assignment is

$$\tilde{s}_{tk}(i,j) = \begin{cases} \sigma(\lambda_e g_{tk}(i,j)) \prod_{k'=1}^{k-1} \sigma(-\lambda_e g_{tk'}(i,j)), & 1 \le k < K \\ \prod_{k'=1}^{K-1} \sigma(-\lambda_e g_{tk'}(i,j)), & k = K. \end{cases} \tag{4}$$

Using the property of the logistic function $\sigma'(x) = \sigma(x)\sigma(-x)$, we can obtain its gradient w. r. t. the hidden field

$$\frac{\partial \tilde{s}_{tk}(i,j)}{\partial g_{tl}(i,j)} = \begin{cases} 0, & k < l \\ \lambda_e \tilde{s}_{tk}(i,j)\sigma(-\lambda_e g_{tl}(i,j)), & k = l \\ -\lambda_e \tilde{s}_{tk}(i,j)\sigma(\lambda_e g_{tl}(i,j)), & k > l. \end{cases} \tag{5}$$

Note that the $k < l$ case means that the hidden field of a layer behind does not influence the data term of the layers in front of it.

It is straightforward to obtain the gradient of the data term at time t w. r. t. the hidden field $\mathbf{g}_{tl}$ as

$$\frac{\partial E_{\text{data}}(\mathbf{u}_t, \mathbf{v}_t, \mathbf{g}_t, \mathbf{g}_{t+1})}{\partial g_{tl}(i,j)} = \sum_k \Big(\rho_d(I_t^s(i,j) - I_{t+1}^s(i + u_{tk}^{ij}, j + v_{tk}^{ij})) - \lambda_d\Big) \frac{\partial \tilde{s}_{tk}(i,j)}{\partial g_{tl}(i,j)} \tilde{s}_{t+1,k}(i + u_{tk}^{ij}, j + v_{tk}^{ij}). \quad (6)$$

The formula for the data term at time $t-1$ is a little more complicated because it depends on the soft layer assignment at time t warped by the flow field. To simplify the notation, we define $h_{tk}(i,j) = \Big(\rho_d(I_t^s(i,j) - I_{t+1}^s(i + u_{tk}^{ij}, j + v_{tk}^{ij})) - \lambda_d\Big)\tilde{s}_{tk}(i,j)$ and rewrite the data term at time $t-1$ in the vector product form as

$$E_{\text{data}}(\mathbf{u}_{t-1}, \mathbf{v}_{t-1}, \mathbf{g}_{t-1}, \mathbf{g}_t) = \mathbf{h}_{t-1,k}(:)^T \mathbf{W}_{t-1,k} \tilde{\mathbf{s}}_{tk}(:), \quad (7)$$

where the warping operator $\mathbf{W}_{t-1,k}$ is the same as above.

Now we can obtain the gradient of the data term w. r. t. the hidden field $\mathbf{g}_{tk}$ as

$$\nabla_{\mathbf{g}_{tk}} E_{\text{data}}(\mathbf{u}_{t-1}, \mathbf{v}_{t-1}, \mathbf{g}_{t-1}, \mathbf{g}_t) = \nabla_{\mathbf{g}_{tk}} \mathbf{s}_{tk}(:) \mathbf{W}_{t-1,k}^T \mathbf{h}_{t-1,k}(:). \quad (8)$$

Note that $\nabla_{\mathbf{g}_{tk}} \tilde{\mathbf{s}}_{tk}(:)$ is a diagonal matrix because $\frac{\partial \tilde{s}_{tk}(i,j)}{\partial g_{tk}(i',j')} = 0$ for $(i,j) \neq (i',j')$.

Color-modulated Spatial Term It is easy to obtain the gradient of E_{space} w. r. t. the hidden field because of its quadratic form:

$$\frac{\partial E_{\text{space}}(\mathbf{g}_{tk})}{\partial g_{tk}(i,j)} = \sum_{(i',j') \in \Gamma(i,j)} 2 w_{i'j'}^{ij} (g_{tk}(i,j) - g_{tk}(i',j')). \quad (9)$$

4.2 Gradients w. r. t. the Horizontal Flow Field

Due to symmetry, we only give the gradient formulas for the horizontal flow field; the vertical case is analogous.

Temporal Coherence Term Using the chain rule, we obtain

$$\frac{\partial E_{\text{time}}(\mathbf{g}_{t,k}, \mathbf{g}_{t+1,k}, \mathbf{u}_{tk}, \mathbf{v}_{tk})}{\partial u_{tk}^{ij}} = 2\big(g_{t+1,k}(i + u_{tk}^{ij}, j + v_{tk}^{ij}) - g_{tk}(i,j)\big) \frac{\partial g_{t+1,k}}{\partial x}(I + u_{tk}^{ij}, j + v_{tk}^{ij}) \quad (10)$$

where $\partial g_{t+1,k} / \partial x$ is the partial derivative of the hidden field in the horizontal image direction x.

Data Term The data term is different from the standard data term for optical flow estimation in that the warped soft layer assignment $\tilde{s}_{t+1,k}(i + u_{tk}^{ij}, j + v_{tk}^{ij})$ also depends on the flow field. As a result

$$\begin{aligned}\frac{\partial E_{\text{data}}(\mathbf{u}_t, \mathbf{v}_t, \mathbf{g}_t, \mathbf{g}_{t+1})}{\partial u_{tk}^{ij}} = \\ - \rho_d'(\mathbf{I}_t^s(i,j) - \mathbf{I}_{t+1}^s(i + u_{tk}^{ij}, j + v_{tk}^{ij})) \frac{\partial \mathbf{I}_{t+1}^s}{\partial x}(i + u_{tk}^{ij}, j + v_{tk}^{ij}) \tilde{s}_{tk}(i,j) \tilde{s}_{t+1,k}(i + u_{tk}^{ij}, j + v_{tk}^{ij}) \\ + \Big(\rho_d(\mathbf{I}_t^s(i,j) - \mathbf{I}_{t+1}^s(i + u_{tk}^{ij}, j + v_{tk}^{ij})) - \lambda_d\Big) \tilde{s}_{tk}(i,j) \frac{\partial \tilde{s}_{t+1,k}}{\partial x}(i + u_{tk}^{ij}, j + v_{tk}^{ij}). \quad (11)\end{aligned}$$

Again the partial derivatives with respect to x correspond to partials in the horizontal image direction.

Spatial Prior Term Before each warping step, we estimate the affine flow field $(\bar{\mathbf{u}}_{\theta_{tk}}, \bar{\mathbf{v}}_{\theta_{tk}})$ for each layer. We solve for the parameters θ_{tk} using a simple least squares estimate given the current flow field for the layer. This could be improved by deriving the partial derivatives of the affine parameters w. r. t. the robust formulation and solving for them along with the other parameters. This is future work.

With the affine flow field fixed, we can obtain the gradient of the spatial term w. r. t. the flow field as

$$\frac{\partial E_{\text{aff}}(\mathbf{u}_{tk}, \theta_{tk})}{\partial u_{tk}^{ij}} = \sum_{(i',j') \in \Gamma(i,j)} \rho_s' \Big((u_{tk}^{ij} - \bar{u}_{\theta_{tk}}^{ij}) - (u_{tk}^{i'j'} - \bar{u}_{\theta_{tk}}^{i'j'}) \Big). \tag{12}$$

With these gradient formulas, it is straightforward to perform the incremental estimation for the flow field [2].

References

[1] C. Liu, W. T. Freeman, E. H. Adelson, and Y. Weiss. Human-assisted motion annotation. In *CVPR*, 2008.

[2] S. Roth and M. J. Black. On the spatial statistics of optical flow. *IJCV*, 74(1):33–50, August 2007.

[3] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268, 1992.

[4] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *CVPR*, 2010.

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

[6] A. Wedel, T. Pock, C. Zach, D. Cremers, and H. Bischof. An improved algorithm for TV-L1 optical flow. In *Proc. of the Dagstuhl Motion Workshop*, LNCS. Springer, September 2008.

[7] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *CVPR*, pages 520–526, Jun 1997.