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

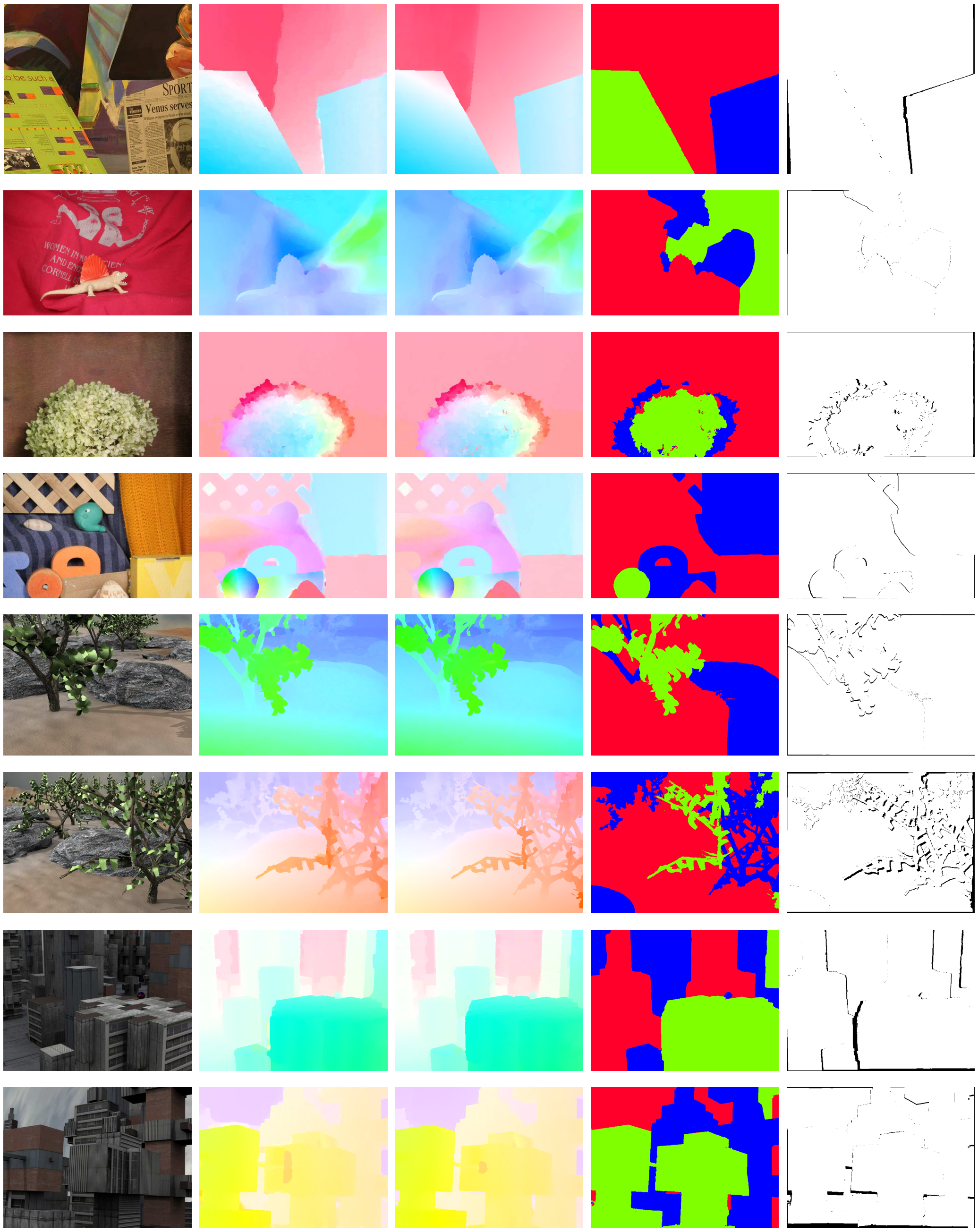


Figure 1: Results on the Middlebury *training* set. Left to right: first image, initial flow field given by Classic+NL, final flow field, motion segmentation, and detected occlusions (black). Best viewed in color and better enlarged for comparing the flow fields.

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

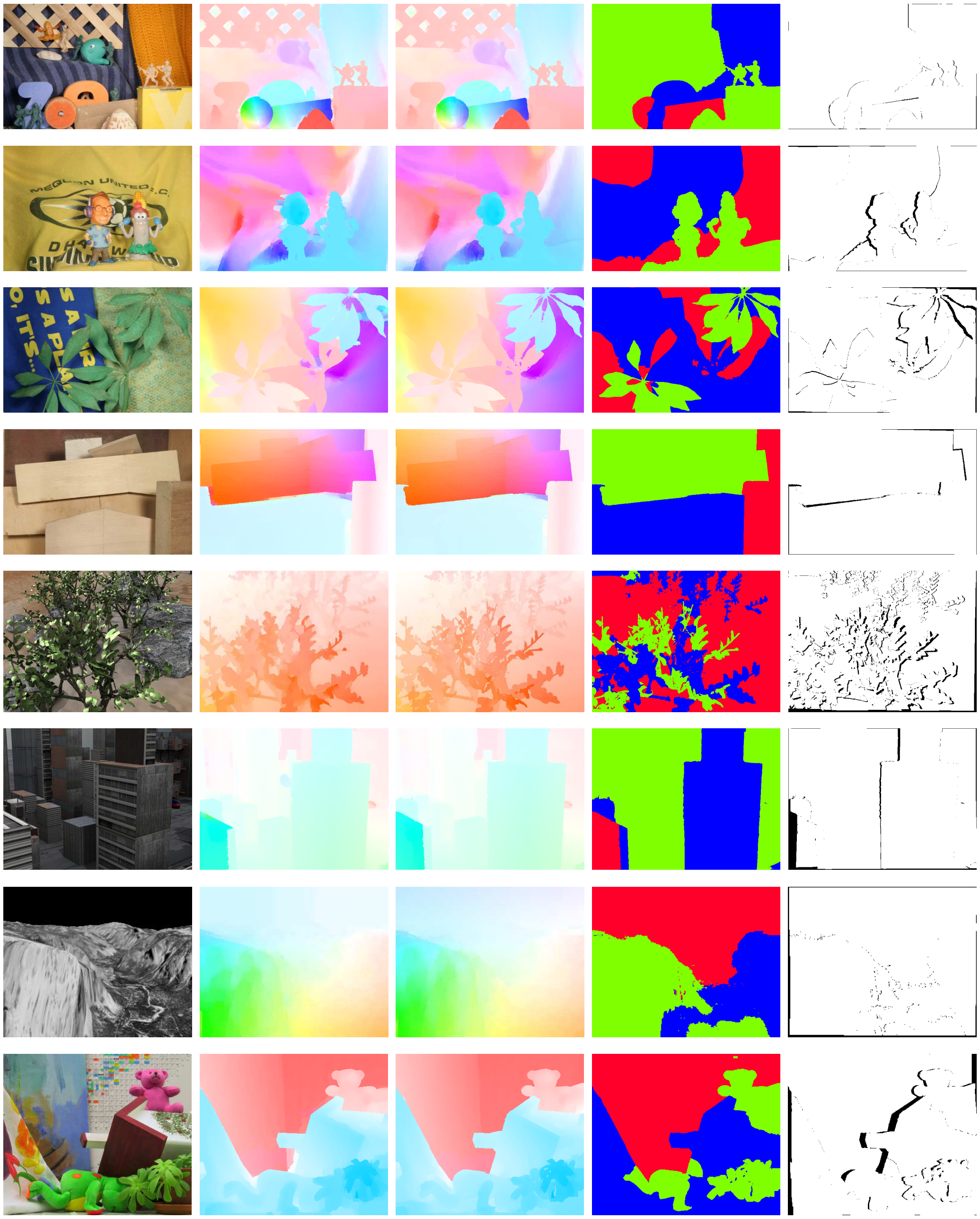
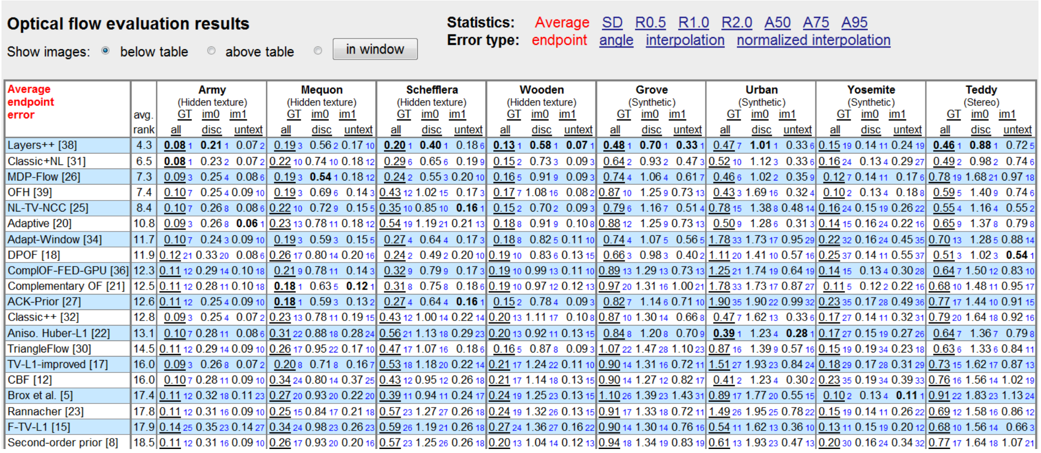


Figure 2: Results on the Middlebury *test* set. Left to right: first image, initial flow field given by Classic+NL, final flow field, motion segmentation, and detected occlusions (black). Best viewed in color and better enlarged for comparing the flow fields.

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.



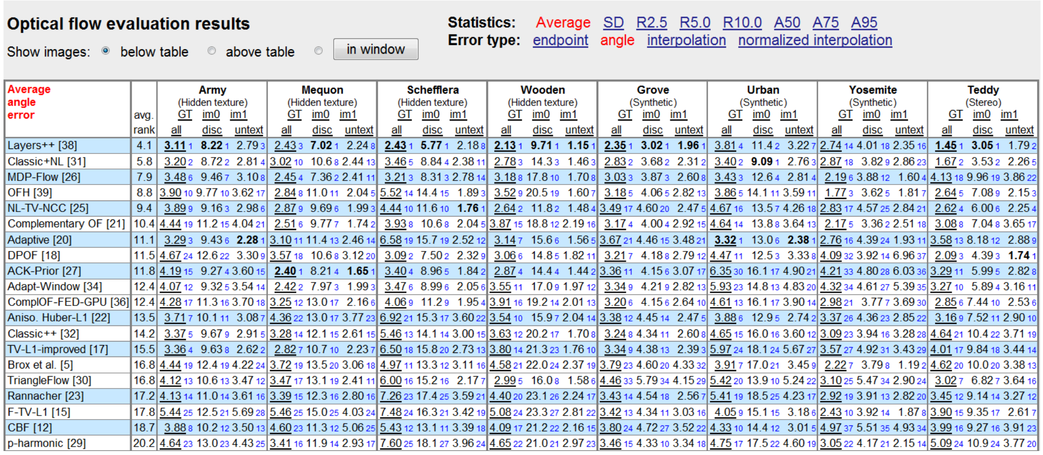
Optical flow evaluation results

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 9	0.12 7	0.14 11	0.17 6	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 6	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 6	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 5	1.78 33	1.73 17	0.95 29	0.22 32	0.16 24	0.45 35	0.70 13	1.28 5	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 6	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 25	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 6	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 16	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

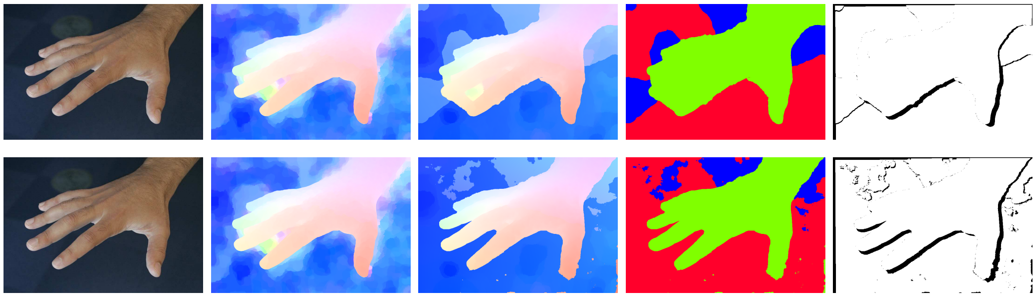


Figure 5: Results on the "Hand" sequence. Top: using same parameters as those used for the Middlebury data set (EPE 2.754). Bottom: using parameters tuned for hand (EPE 1.909). Left to right: first image, initial flow field given by Classic+NL, final flow field, motion segmentation, and detected occlusions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

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

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 $h_{tk}(i,j) = \Big(\rho_d(I^s_t(i,j) - I^s_{t+1}(i+u^{ij}_{tk}, j+v^{ij}_{tk})) - \lambda_d\Big)\tilde{s}_{tk}(i,j)$ 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

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

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:

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

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

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

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 $\tilde{s}_{t+1,k}(i+u^{ij}_{tk}, j+v^{ij}_{tk})$ also depends on the flow field. As a result

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

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.