A Quantitative Analysis of Current Practices in Optical Flow Estimation and the Principles Behind Them

Deqing Sun, Stefan Roth and Michael J. Black

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-10-03
January 2013

A Quantitative Analysis of Current Practices in Optical Flow Estimation and the Principles Behind Them

Deqing Sun · Stefan Roth · Michael J. Black

the date of receipt and acceptance should be inserted later

Abstract The accuracy of optical flow estimation algorithms has been improving steadily as evidenced by results on the Middlebury optical flow benchmark. The typical formulation, however, has changed little since the work of Horn and Schunck. We attempt to uncover what has made recent advances possible through a thorough analysis of how the objective function, the optimization method, and modern implementation practices influence accuracy. We discover that "classical" flow formulations perform surprisingly well when combined with modern optimization and implementation techniques. One key implementation detail is the median filtering of intermediate flow fields during optimization. While this improves the robustness of classical methods it actually leads to higher energy solutions, meaning that these methods are not optimizing the original objective function. To understand the principles behind this phenomenon, we derive a new objective function that formalizes the median filtering heuristic. This objective function includes a non-local smoothness term that robustly integrates flow estimates over large spatial neighborhoods. By modifying this new term to include information about flow and image boundaries we develop a method that can better preserve motion details.

Deqing Sun
Department of Computer Science, Harvard University, Cambridge, MA, USA
e-mail: dqsun@seas.harvard.edu
The work for this paper was performed while DS was at Brown University

Stefan Roth
Department of Computer Science, TU Darmstadt, Darmstadt, Germany
e-mail: sroth@cs.tu-darmstadt.de

Michael J. Black
Max Planck Institute for Intelligent Systems, Tübingen, Germany
e-mail: black@is.mpg.de

Keywords Optical flow estimation · Practices · Median filtering · Non-local term · Motion boundary

1 Introduction

The field of optical flow estimation is making steady progress as evidenced by the increasing accuracy of current methods on the Middlebury optical flow benchmark [6]. After nearly 30 years of research, these methods have obtained an impressive level of reliability and accuracy [48,49,51,55,59]. *But what has led to this progress?* The majority of today's methods strongly resemble the original formulation of Horn and Schunck (**HS**) [22]. They combine a data term that assumes constancy of some image property with a spatial term that models how the flow is expected to vary across the image. An objective function combining these two terms is then optimized. Given that this basic structure is unchanged since **HS**, what has enabled the performance gains of modern approaches?

The paper has three parts. In the first, we perform a study of current optical flow methods and models. The most accurate methods on the Middlebury flow dataset make different choices about how to model the objective function, how to approximate this model to make it computationally tractable, and how to optimize it. Since most published methods change *all* of these properties at once, it can be difficult to know which choices are most important. To address this, we define a baseline algorithm that is "classical", in that it is a direct descendant of the original **HS** formulation, and then systematically vary the model and method using different techniques from the art. The results are surprising. We find that only a small number of key choices produce statistically significant improvements and that they can be combined into a very simple method that achieves accura-

cies near the state of the art. More importantly, our analysis reveals what makes current flow methods work so well.

Part two examines the *principles* behind this success. We find that one algorithmic choice produces the most significant improvements: applying a median filter to intermediate flow values during incremental estimation and warping [48, 49]. While this heuristic improves the accuracy of the recovered flow fields, it actually *increases* the energy of the objective function. This suggests that what is being optimized is actually a new and different objective. Using observations about median filtering and L1 energy minimization from Li and Osher [29], we formulate a new *non-local term* that is added to the original, classical objective. This new term goes beyond standard local (pairwise) smoothness to robustly integrate information over large spatial neighborhoods. We show that minimizing this new energy approximates the original optimization with the heuristic median filtering step. Note, however, that the new objective falls outside our definition of classical methods.

Finally, once the median filtering heuristic is formulated as a non-local term in the objective, we immediately recognize how to modify and improve it. In part three we show how information about image structure and flow boundaries can be incorporated into a weighted version of the non-local term to prevent over-smoothing across boundaries. By incorporating structure from the image, this weighted version does not suffer from some of the errors produced by median filtering and better preserves motion boundaries. Figure 1 illustrates optical flow estimates for a range of methods from a "basic" **HS** method to our newly proposed **Classic+NL** method.

In summary, the contributions of this paper are to (1) analyze current flow models and methods to understand which design choices matter; (2) formulate and compare several classical objectives descended from **HS** using modern methods; (3) formalize one of the key heuristics and derive a new objective function that includes a non-local term; (4) modify this new objective to produce a state-of-the-art method. In doing this, we provide a "recipe" for others studying optical flow that can guide their design choices. Finally, to enable comparison and further innovation, we provide a public MATLAB implementation [1].

At the time of writing our conference paper [41] (March 2010), the resulting approach was ranked 1st in both angular and end-point errors in the Middlebury evaluation. At the writing of the paper (Sep. 2012), the method, **Classic+NL**, ranks 13th in both AAE and EPE. Several recent and high-ranking methods directly build on **Classic+NL**, such as layered models [43], methods with more advanced motion prior models [16, 24], and efficient optimization schemes for the non-local term [26]. Compared to the conference version [41], this paper includes many more detailed results of both the basic and the improved models.

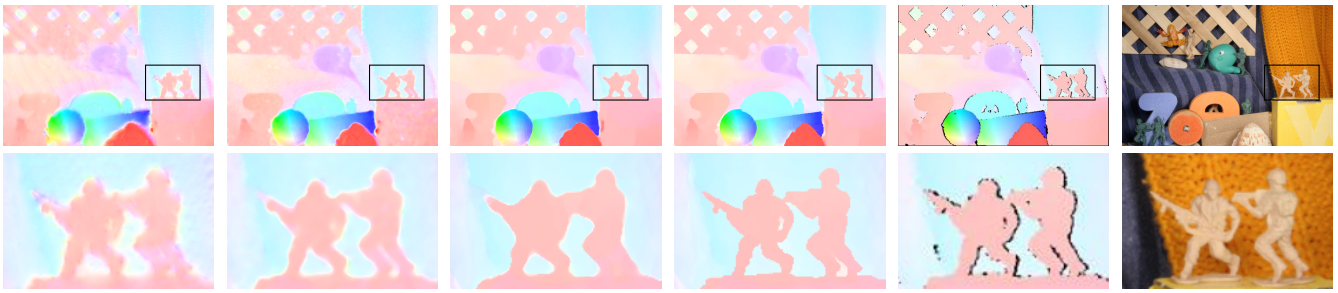
2 Previous Work

It is important to separately analyze the contributions of the objective function that defines the problem (*the model*) and the optimization algorithm and implementation used to minimize it (*the method*). The **HS** formulation, for example, has long been thought to be highly inaccurate. Barron *et al.* [7] reported an average angular error (AAE) of ~30 degrees on the "Yosemite" sequence. This confounds the objective function with the particular optimization method proposed by Horn and Schunck. Horn and Schunck noted that the correct way to optimize their objective is by solving a system of linear equations as is common today. This was impractical on the computers of the day, hence they used a heuristic method. In factm Barron *et al.* note that the original **HS** derivatives were implemented crudely and report a modified version of **HS** with AAE around 11 degrees. When optimized with today's methods, the **HS** objective achieves surprisingly competitive results despite the expected over-smoothing and sensitivity to outliers [17]. The reported accuracy of a method is jointly determined by the objective function, the optimization techniques, the implementation details, and the parameter tuning/learning (cf. [33, 44]). We review papers in the context of the first three aspects below.

Models: The global formulation of optical flow introduced by Horn and Schunck [22] relies on both brightness constancy and spatial smoothness assumptions, but suffers from the fact that the quadratic formulation is not robust to outliers. Black and Anandan [10] address this by replacing the quadratic error function with a robust formulation. Subsequently, many different robust functions have been explored [12, 28, 42] and it remains unclear which is best. We refer to all these spatially-discrete formulations derived from **HS** as "classical." We systematically explore variations in the formulation and optimization of these approaches. The surprise is that the classical model, appropriately implemented, remains fairly competitive.

There are many formulations beyond the classical ones that we do not consider here. Significant ones use oriented smoothness [34, 42, 48, 58, 59], rigidity constraints [47, 48], or image segmentation [9, 27, 54, 60]. While they deserve similar careful consideration, we expect many of our conclusions to carry forward. Note that one can select among a set of models for a given sequence [32], instead of finding a "best" model for all the sequences.

Methods: Many of the implementation details that are thought to be important date back to the early days of optical flow. Current best practices include coarse-to-fine estimation to deal with large motions [8, 13], texture decomposition [47, 49] or high-order filter constancy [4, 12, 20, 28, 59] to reduce the influence of lighting changes, incremental warping [8], warping with bicubic interpolation [28, 49], temporal averaging of image derivatives [21, 49], graduat-



(a) "Old" **HS** [42] (b) "New" **HS** (c) **Classic++** (d) **Classic+NL** (e) Ground truth (f) First frame

Fig. 1 Estimated optical flow on the Middlebury test "Army" sequence. Left to right: (a) an old implementation of the Horn & Schunck (**HS**) method [42], (b) a new implementation with current practices, (c) a modern implementation of a robust version, (d) an improved model that uses a non-local spatial term to robustly integrate information over a large spatial neighborhood, (e) ground truth from the Middlebury website (the quality of the "ground truth" is lower than the actual one because of compression), and (f) the first frame. Color coding as in [6], shown in Fig. 4 (c).

ed non-convexity [11] to minimize non-convex energies [10, 42], and median filtering after each incremental estimation step to remove outliers [49].

This median filtering heuristic is of particular interest as it makes non-robust methods more robust and improves the accuracy of all methods we tested. The effect on the objective function and the underlying reason for its success have not previously been analyzed. Least median squares estimation can be used to robustly reject outliers in flow estimation [5], but previous work has focused on the data term.

Related to median filtering, and our new non-local term, is the use of bilateral filtering to prevent smoothing across motion boundaries [53]. This approach separates a variational method into two filtering update stages, and replaces the original anisotropic diffusion process with multi-cue driven bilateral filtering. As with median filtering, the bilateral filtering step changes the original energy function.

Models that are formulated with an L1 robust penalty are often coupled with specialized total variation (TV) optimization methods [57]. Here we focus on generic optimization methods that can apply to any model and find that the estimated flow fields are as accurate as the reported results for specialized methods.

Despite recent algorithmic advances, there is a lack of publicly available, easy to use, and accurate flow estimation software. The GPU4Vision project [2] has made a substantial effort to change this and provides executable files for several accurate methods [47, 48, 49, 51]. The dependence on the GPU and the lack of source code are limitations. We hope that our public MATLAB code will not only help in understanding the practices of optical flow, but also let others exploit optical flow as a useful tool in computer vision and related fields.

3 Classical Models

As is common to "classical" methods we only address the two-frame optical flow estimation problem. We write the classical optical flow objective function in its spatially discrete form as

$$E(\mathbf{u},\mathbf{v}) = \sum_{i,j} \{\rho_D(I_1(i,j) - I_2(i+u_{i,j}, j+v_{i,j})) \quad (1)$$
$$+\lambda[\rho_S(u_{i,j} - u_{i+1,j}) + \rho_S(u_{i,j} - u_{i,j+1})$$
$$+\rho_S(v_{i,j} - v_{i+1,j}) + \rho_S(v_{i,j} - v_{i,j+1})]\},$$

where $\mathbf{u}$ and $\mathbf{v}$ are the horizontal and vertical components of the optical flow field to be estimated from images I_1 and I_2, λ is a regularization parameter, and ρ_D and ρ_S are the data and spatial penalty functions. We consider three different penalty functions: (1) the quadratic **HS** penalty $\rho(x) = x^2$; (2) the Charbonnier penalty $\rho(x) = \sqrt{x^2 + \epsilon^2}$ [13], a differentiable variant of the absolute value, the most robust convex function; and (3) the Lorentzian $\rho(x) = \log(1 + \frac{x^2}{2\sigma^2})$, which is a non-convex robust penalty used in [10]. We refer to the robust formulation with the Lorentzian penalty as **BA** (short for Black and Anandan). Note that this classical model is related to a standard pairwise Markov random field (MRF) based on a 4-neighborhood [18].

In the remainder of this section we define a baseline method using several techniques from the literature. This is not the "best" method, but includes modern techniques and will be used for comparison. We only briefly describe the main choices, which are explored in more detail in the following section and the cited references.

Quantitative results are presented throughout the remainder of the text. In all cases we report the average end-point error (EPE) on the Middlebury training and test sets, depending on the experiment. Given the extensive nature of the evaluation, only average results are presented in the main body, while the details for each individual sequence are provided at the end of the paper.

3.1 Baseline methods

To gain robustness against lighting changes, we follow [49] and apply the Rudin-Osher-Fatemi (ROF) structure texture

decomposition method [38] to pre-process the input sequences and linearly combine the texture and structure components (in the proportion 20:1). The parameters are set according to [49].

Optimization is performed using a standard incremental multi-resolution technique (e. g., [10, 13]) to estimate flow fields with large displacements. The optical flow estimated at a coarse level is used to warp the second image toward the first at the next finer level, and a flow increment is calculated between the first image and the warped second image. The standard deviation of the Gaussian anti-aliasing filter is set to be $\frac{1}{\sqrt{2d}}$, where d denotes the downsampling factor. Each level is recursively downsampled from its nearest lower level. In building the pyramid, the downsampling factor is not critical as pointed out in the next section and here we use the settings in [42], which uses a factor of 0.8 in the final stages of the optimization. We adaptively determine the number of pyramid levels so that the top level has a width or height of around 20 to 30 pixels. At each pyramid level, we perform 10 warping steps to compute the flow increment.

At each warping step, we linearize the data term once, which involves computing terms of the type $\frac{\partial}{\partial x} I_2(i+u^k_{i,j}, j+v^k_{i,j})$, where $\partial/\partial x$ denotes the partial derivative in the horizontal direction, u^k and v^k denote the current flow estimate at iteration k. As suggested in [49], we compute the derivatives of the second image using the 5-point derivative filter $\frac{1}{12}[-1\ 8\ 0\ -8\ 1]$, and warp the second image and its derivatives toward the first using the current flow estimate by bicubic interpolation. We then compute the spatial derivatives of the first image, compute the average of these and the corresponding warped derivatives of the second image (cf. [21]), and use these in place of $\frac{\partial I_2}{\partial x}$. For pixels moving out of the image boundaries, we set both their corresponding temporal and spatial derivatives to zero. After each warping step, the flow update is computed, and then we apply a 5×5 median filter to the newly computed flow field to remove outliers [49].

For the Charbonnier (**Classic-C**) and Lorentzian (**Classic-L**) penalty function, we use a graduated non-convexity (GNC) scheme [11] as described in [42]. First, we replace the robust penalty functions by quadratic penalty functions and obtain a quadratic formulation of the objective function, $E_Q(\mathbf{u}, \mathbf{v})$. Then we linearly combine the quadratic penalty function with the desired robust penalty function and gradually change the weighting of the two terms to reach the desired robust penalty function. In practice, we use a three-stage GNC scheme, with the objective functions for the first, second, and third stages being $E_Q(\mathbf{u}, \mathbf{v})$, $\frac{1}{2}\big(E_Q(\mathbf{u}, \mathbf{v}) + E(\mathbf{u}, \mathbf{v})\big)$, and $E(\mathbf{u}, \mathbf{v})$ respectively. The output of a previous stage serves as the initialization to the next stage. The standard deviations of the corresponding quadratic penalty function are set to be 1 for the Charbonnier penalty and, for the Lorentzian, are taken to be the same as the σ value used in the Lorentzian function. The same regularization weight λ is used for both the quadratic and the robust objective functions.

	Avg. Rank	Avg. EPE
Classic-C	34.8	0.408
HS	49.0	0.501
Classic-L	42.7	0.530
Classic-C-brightness	N/A	0.726
HS-brightness	N/A	0.759
Classic-L-brightness	N/A	0.603
HS [42]	66.2	0.872
BA (Classic-L) [42]	59.6	0.746
Adaptive [48]	28.5	0.401
Complementary OF [59]	31.6	0.485

Table 1 Models. Average rank and end-point error (EPE) on the Middlebury *test set* using different penalty functions. Two state-of-the-art methods in Dec. 2010 are included for comparison. The ranking information was obtained at the writing of the paper (Sep. 2012). Please refer Table 11 for the EPE results on each sequence.

3.2 Baseline results

The regularization parameter λ is selected among a set of candidate values to achieve the best average end-point error (EPE) on the Middlebury training set. For the Charbonnier penalty function, the candidate set is $[1,\ 3,\ 5,\ 8,\ 10]$ and 5 is optimal. The Charbonnier penalty uses $\epsilon = 0.001$ for both the data and the spatial term in Eq. (1). The Lorentzian uses $\sigma = 1.5$ for the data term, $\sigma = 0.03$ for the spatial term, and $\lambda = 0.06$. These parameters are fixed throughout the experiments, except where mentioned.

Table 1 summarizes the EPE results of the basic model with three different penalty functions on the Middlebury test set, along with the two top performers at the time of performing the evaluation (considering only published papers when the evaluation table was generated). The classic formulations with two non-quadratic penalty functions (**Classic-C**) and (**Classic-L**) achieve competitive results despite their simplicity. The baseline optimization of **HS** and **BA** (**Classic-L**) results in significantly better accuracy than previously reported for these models [42]. Note that the analysis also holds for the training set (Table 2).

Because **Classic-C** performs quite well despite its simplicity, we set it as the baseline below. Note that our baseline implementation of **HS** has a lower average EPE than many more sophisticated methods. The **HS** implementation here incorporates many algorithmic and implementation details not present in the original **HS** method; the core idea of quadratic data and spatial terms however remains the same. In our naming convention, one can think of the **HS** method here as **Classic-Q**, meaning that it is the same as the **Classic-C** method except that the data and spatial penalty terms are quadratic.

	Avg. EPE	significance	p-value
Classic-C	0.298	—	—
HS	0.384	**1**	**0.0078**
Classic-L	0.319	**1**	**0.0078**
Classic-C-brightness	0.288	0	0.9453
HS-brightness	0.387	**1**	**0.0078**
Classic-L-brightness	0.325	0	0.2969
Gradient	0.305	0	0.4609
Gaussian + Dx + Dy	0.290	0	0.6406
Sobel edge magnitude [45]	0.417	**1**	**0.0156**
Laplacian [28]	0.430	**1**	**0.0078**
Laplacian1:1	0.301	0	0.6641
Gaussian pre-filtering ($\sigma = 0.5$)	0.281	0	0.5469
Texture4:1	0.286	0	0.5312
Unnormalized texture	0.298	0	0.3750

Table 2 Pre-Processing. Average end-point error (EPE) on the Middlebury *training set* for the baseline method (**Classic-C**) using different image pre-processing techniques. Significance is always with respect to **Classic-C**. Please refer to Tables 12 and 13 for the detailed results on each training sequence.

4 Practices Explored

We now systematically vary the baseline approach by incorporating different ideas that have appeared in the literature, with the goal of illuminating which of these ideas are significant. This analysis is performed on the Middlebury training set by changing only *one property at a time*. Statistical significance is determined using a Wilcoxon signed rank test [52] between each modified method and the baseline **Classic-C** method; a p value less than 0.05 indicates a significant difference. Each section below presents detailed comparisons of all these methods and then summarizes the results in a simple "take away message" about what we think are the "best practices" based on the data.

4.1 Image Pre-Processing

While it is common to talk about the brightness constancy assumption as a core feature of most optical flow algorithms, in practice many other constancy assumptions have been used. It is common, for example, to pre-filter the images in a variety of ways ranging from simple smoothing to edge detection. For each method, we optimize the regularization parameter λ for the training sequences. The results are summarized in Table 2, with details of the methods applied to individual training sequences given in Tables 12 and 13. The baseline uses a non-linear pre-filtering of the images (ROF) to reduce the influence of illumination changes between frames [49]. Table 2 shows the effect of using no pre-processing, resulting in the standard brightness constancy model (***-brightness**). **Classic-C-brightness** actually achieves lower EPE on the training set than does **Classic-C** but significantly higher error on the test set (Table 1). This disparity suggests overfitting to the training data and leaves open the question as to whether the standard brightness constancy assumption, formulated robustly, may still compete with various types of filter/structure constancy given appropriate training data.

Simpler alternatives, such as filter response (or high-order) constancy [12,42] can serve the same purpose as ROF texture decomposition. A variety of pre-filters have been used in the literature, including derivative filters, Laplacians [15, 28], and Gaussians. Edges have also been emphasized using the Sobel edge magnitude [45].

Gradient only imposes constancy of the gradient vector at each pixel as proposed in [12]; i. e., it robustly penalizes Euclidean distance between image gradients. We use central difference filters ($Dx = [-0.5\ 0\ 0.5]$ and $Dy = Dx^T$). **Gaussian+Dx+Dy** assumes separate brightness, horizontal derivative, and vertical derivative constancy. A weighted combination of robust functions applied to each term is used as in [42]. Neither of these methods differ significantly from the baseline texture decomposition (**Classic-C**). Two methods are significantly worse: the **Sobel edge magnitude** [45] and **Laplacian** pre-filtering (5×5) as used in [28]. **Sobel edge magnitude** appears to not work well on some of the sequences, particularly the synthetic ones, and may not be suitable for a general flow estimation method. **Laplacian** pre-filtering (5×5) as used in [28] produces good results on "RubberWhale", but poor ones on the synthetic sequences. Note that the parameters for the FusionFlow method [28] were mainly tuned using the "RubberWhale" sequence. The evaluation results suggest room for improving the FusionFlow method by a better pre-processing technique. **Gaussian pre-filtering** ($\sigma = 0.5$) performed well on the synthetic sequences, but poorly on real ones. Finally, the texture-structure blending ratio is 20:1 in [49] but 4:1 in [51]. We find that (**Texture**4**:**1) performs better (but not significantly) on the synthetic sequences with a little degradation on the real ones. By default, the blended result from texture decomposition is normalized to $[-1, 1]$ in [49] and $[0, 255]$ in our experiment. Not doing this normalization (**Unnormalized texture**) has little effect.

For the Laplacian pre-filtering, we find combining the filtered image with the original image, in the proportion 1:1, improves accuracy significantly (**Laplacian**1**:**1). Similar to the ROF texture decomposition, such an approach boosts the high frequency while suppressing the low frequency components that contain the lighting change.

Good Practices: Some form of image filtering is useful but simple derivative constancy is nearly as good as the more sophisticated texture decomposition method.

4.2 Coarse-to-Fine Estimation and Graduated Non-Convexity (GNC)

We vary the number of warping steps per pyramid level and find that **3 warping steps** gives similar results as using the

	Avg. EPE	significance	p-value
Classic-C	0.298	—	—
3 **warping steps**	0.304	0	0.9688
Down-0.5	0.298	0	1.0000
w/o GNC	0.354	0	0.1094
Bilinear	0.302	0	0.1016
w/o TAVG	0.306	0	0.1562
Central derivative filter	0.300	0	0.7266
7-point derivative filter [13]	0.302	0	0.3125
Deriv-warp	0.297	0	0.9531
Bicubic-II	0.290	**1**	**0.0391**
Deriv-warp-II	0.287	**1**	**0.0156**
Warp-deriv-II	0.288	**1**	**0.0391**
C-L ($\lambda = 0.6$)	0.303	0	0.1562
L-C ($\lambda = 2$)	0.306	0	0.1562
GC-0.45 ($\lambda = 3$)	0.292	**1**	**0.0156**
GC-0.25 ($\lambda = 0.7$)	0.298	0	1.0000
MF 3×3	0.305	0	0.1016
MF 7×7	0.305	0	0.5625
$2\times$ **MF**	0.300	0	1.0000
$5\times$ **MF**	0.305	0	0.6875
w/o MF	0.352	**1**	**0.0078**
Classic++	0.285	**1**	**0.0078**

Table 3 Model and Methods. Average end-point error (EPE) on the Middlebury *training set* for the baseline method (**Classic-C**) using different algorithm and modeling choices. Please refer to Table 14 for the detailed results on each sequence.

baseline 10 (Table 3), except on "Urban3", which is dominated by large motion and occlusions (see Table 14 for sequence-specific results). For the coarse-to-fine pyramid, [42] uses a downsampling factor of 0.8 during non-convex optimization. A traditional downsampling factor of 0.5 (**Down-**0.5), however, has nearly identical performance. Note that a larger factor means that the pyramid levels are more similar in size and, for a pyramid with top bottom levels of the same size, results in more pyramid levels.

Previously, Brox *et al.* [12] have reported that a downsampling factor of 0.95 produces much better results than 0.5. Note that for each iterative warping estimation step, Brox *et al.* use successive over-relaxation (SOR) to iteratively solve their linear system of equations and stop the iteration before convergence. With a downsampling factor of 0.95, they effectively increase the number of iterative warping steps performed by the algorithm, and this likely helps the overall algorithm converge. For our implementation, we solve the linear system of equations using the MATLAB built-in backslash function and obtain converged results for each iterative warping estimation step. Under such a setting, we find that the downsampling factor has little influence on the performance.

Removing the GNC procedure for the Charbonnier penalty function (**w/o GNC**) results in higher EPE on most sequences and higher energy on all sequences (Table 5). This suggests that the GNC method is helpful even for the convex Charbonnier penalty function due to the nonlinearity of the data term.

Good Practices: The downsampling factor does not matter when using a convex penalty; a standard factor of 0.5 is fine. Some form of GNC is useful even for a convex robust penalty like Charbonnier because of the nonlinear data term.

4.3 Interpolation Method and Derivatives

We find that the baseline bicubic interpolation is more accurate than bilinear (Table 3, **Bilinear**), as already reported in previous work [49]. Removing temporal averaging of the gradients (**w/o TAVG**), using a **Central difference filter** $[-1\ 0\ 1]/2$, or using a 7**-point derivative filter** $[-1\ 9\ -45\ 0\ 45\ -9\ 1]/60$ [13] all reduce accuracy compared to the baseline, but not significantly.

The baseline method computes the image derivative by first computing the derivative of the second image, warping the intermediate result toward the first image, and then averaging the warped result with the spatial derivative of the first image. Another approach is to first warp the second image toward the first image, compute the derivatives of the warped image, and then perform the temporal averaging with the spatial derivatives of the first image [13]. We find the second approach produces similar results (**Deriv-warp**). However, the derivatives computed in either way are inconsistent with those implicitly interpolated by the bicubic interpolation. Bicubic interpolation interpolates not only the image but also the derivatives [36]. Because the MATLAB built-in function *interp2* is based on cubic convolution [25] and does not provide the derivatives used in interpolation, we use the spline-based implementation in [36]. With the new implementation (**Bicubic-II**), the three different ways to compute the derivatives give very similar EPE results, all better than the MATLAB built-in function. However, the one with consistent derivatives (**Bicubic-II**) gives the lowest energy solution, as shown in Table 4.

Good Practices: Use spline-based bicubic interpolation with a 5-point filter. Compute the derivative during the interpolation to obtain the lowest energy solutions. Temporal averaging of the derivatives is probably worthwhile for a small computational expense.

4.4 Penalty Functions

We find that the convex Charbonnier penalty performs better than the more robust, non-convex Lorentzian on both the training and test sets. We test using the Charbonnier for the data term and Lorentzian for the spatial term (**C-L**) and vice versa (**L-C**). The two approaches perform better than using the Lorentzian for both terms but worse than using the Charbonnier for both terms.

One reason might be that non-convex functions are more difficult to optimize, causing the optimization scheme to find a poor local optimum. Another reason might be the MAP

	Sum	Venus	Dimetrodon	Hydrangea	RubberWhale	Grove2	Grove3	Urban2	Urban3
Bicubic-II	8.761	0.552	0.734	0.835	0.481	1.656	2.167	1.061	1.275
Deriv-warp	8.917	0.559	0.745	0.840	0.484	1.682	2.201	1.073	1.333
Warp-deriv	9.035	0.563	0.745	0.845	0.486	1.694	2.238	1.117	1.347

Table 4 Eq. (1) energy ($\times 10^6$) for the optical flow fields computed on the Middlebury *training set*, evaluated using spline-based bicubic interpolation [36]. Note the derivatives consistent with the interpolation method (**Bicubic-II**) produce the lowest energy solution.

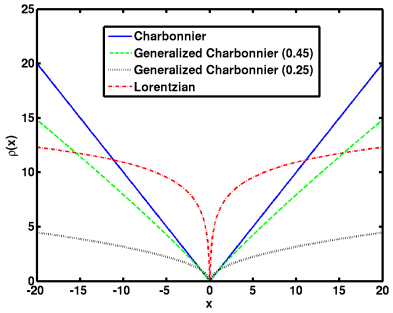


Fig. 2 Different penalty functions for the spatial terms: Charbonnier ($\epsilon = 0.001$), generalized Charbonnier ($a = 0.45$ and $a = 0.25$), and Lorentzian ($\sigma = 0.03$).

estimator actually favors the "wrong" penalty functions [35, 40].

We investigate a generalized Charbonnier penalty function $\rho(x) = (x^2 + \epsilon^2)^a$ that is equal to the Charbonnier penalty when $a = 0.5$, and non-convex when $a < 0.5$ (see Fig. 2). We optimize the regularization parameter λ again. We find a slightly non-convex penalty with $a = 0.45$ (**GC-**0.45) performs consistently better than the Charbonnier penalty, whereas more non-convex penalties (**GC-**0.25 with $a = 0.25$) show no improvement.

Good Practices: The less-robust Charbonnier is preferable to the highly non-convex Lorentzian and a slightly non-convex penalty function (**GC-**0.45) is better still.

4.5 Median Filtering

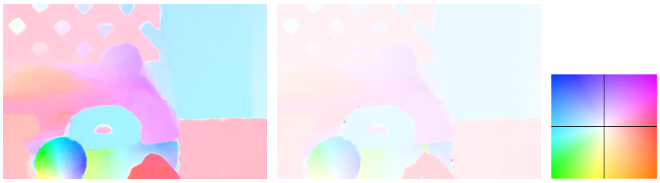
Figure 3 illustrates the median filtering step within the coarse-to-fine incremental estimation process. The baseline 5×5 median filter (**MF** 5×5) is better than both **MF** 3×3 [49] and **MF** 7×7 but the difference is not significant (Table 3). When we perform 5×5 median filtering twice ($2\times$ **MF**) or five times ($5\times$ **MF**) per warping step, the results are worse. Finally, removing the median filtering step (**w/o MF**) makes the computed flow significantly less accurate with larger outliers as shown in Table 3 and Fig. 4.

One interesting result with **HS** is that repeatedly applying median filtering (20 times) at every warping step improves the **HS** formulation and the improvement is statistically significant (**HS** $20\times$ **MF** in Table 17).

Good Practices: Median filtering the intermediate flow results once after every warping iteration is the single most important implementation detail here; 5×5 is a good filter size.

4.6 Best Practices

Combining the analysis above into a single approach means modifying the baseline to use the slightly non-convex generalized Charbonnier and the spline-based bicubic interpolation. This leads to a statistically significant improvement over the baseline (Table 3, **Classic++**). This method is directly descended from **HS** and **BA**, yet updated with the current best optimization practices known to us. This simple method ranks 32^{th} out of 73 methods in both EPE and AAE on the Middlebury test set at the writing of the paper (Sep. 2012). However, as we will see soon, this method is somehow not "simple". Instead of the original objective, a different objective is being optimized with the median filtering step. The same is true for the reported results of both **HS** and **BA**.



(a) **With** median filtering (b) **Without** median filtering (c) Key

Fig. 4 Estimated flow fields on sequence "RubberWhale" using **Classic-C** with and without (**w/o MF**) the median filtering step. (a) (**w/ MF**) energy $502,387$, (b) (**w/o MF**) energy $449,290$, (c) color key [6]. The median filtering step helps reach a solution free from outliers but with a higher energy.

5 Models Underlying Median Filtering

Our analysis reveals the practical importance of median filtering during optimization. This effectively *denoises* the intermediate flow fields, preventing gross outliers, and making even non-robust methods like **HS** more robust. We ask whether there is a *principle* underlying this heuristic?

One interesting observation is that flow fields obtained with median filtering have substantially *higher* energy than those without (Table 5 and Fig. 4). If the median filter is helping to optimize the objective, it should lead to lower

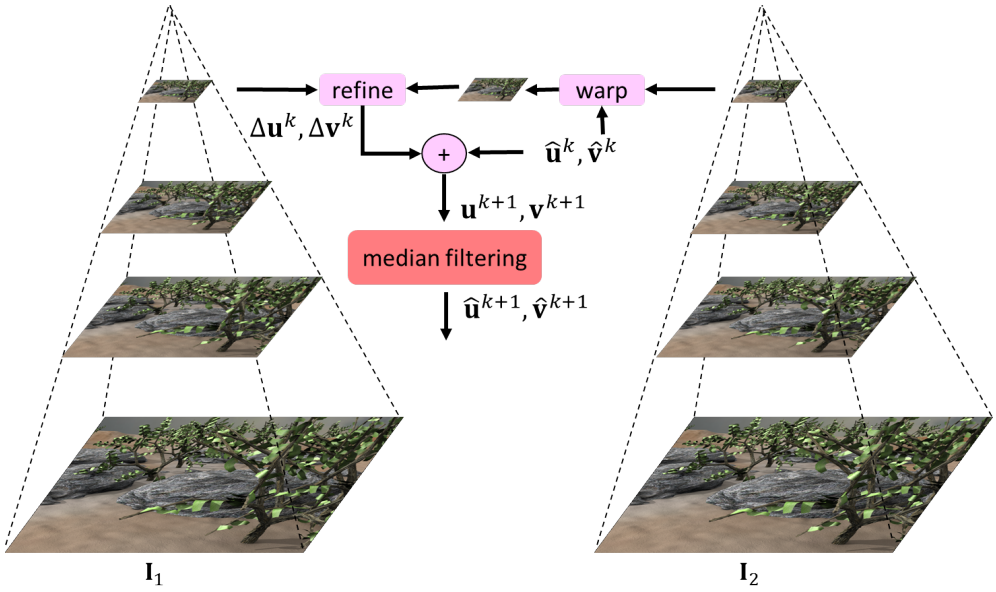


Fig. 3 The median filtering is performed after every incremental warping step (i. e., once at every image pyramid level). The output of the median filtering is upsampled and used as the initial estimate for the next larger pyramid level.

	Sum	Venus	Dimetrodon	Hydrangea	RubberWhale	Grove2	Grove3	Urban2	Urban3
Classic-C	9.388	0.589	0.748	0.866	0.502	1.816	2.317	1.126	1.424
w/o GNC	9.689	0.593	0.750	0.870	0.506	1.845	2.518	1.142	1.465
w/o MF	8.044	0.517	0.701	0.668	0.449	1.418	1.830	1.066	1.395

Table 5 Eq. (1) energy ($\times 10^6$) for the optical flow fields computed on the Middlebury *training set*, evaluated using convolution-based bicubic interpolation [25]. Note that **Classic-C** uses graduated non-convexity (GNC), which reduces the energy, and median filtering, which increases it.

energies. Higher energies and more accurate estimates suggest that incorporating median filtering changes the objective function being optimized.

The insight that follows from this is that the median filtering heuristic is related to the minimization of an objective function that differs from the classical one. In particular the optimization of Eq. (1), with interleaved median filtering, approximately minimizes

$$E(\mathbf{u},\mathbf{v}) = \tag{2}$$
$$\sum_{i,j}\Big\{\rho_D(I_1(i,j)-I_2(i+u_{i,j},j+v_{i,j}))$$
$$+\lambda[\rho_S(u_{i,j}-u_{i+1,j})+\rho_S(u_{i,j}-u_{i,j+1})$$
$$+\rho_S(v_{i,j}-v_{i+1,j})+\rho_S(v_{i,j}-v_{i,j+1})]\Big\}$$
$$+\lambda_N\sum_{i,j}\sum_{(i',j')\in\mathcal{N}_{i,j}}(|u_{i,j}-u_{i',j'}|+|v_{i,j}-v_{i',j'}|),$$

where $\mathcal{N}_{i,j}$ is the set of neighbors of pixel (i,j) in a possibly large area and λ_N is a scalar weight. The term in braces is the same as the flow energy from Eq. (1), while the last term is new. This *non-local* term [14, 19] imposes a particular smoothness assumption within a specified region of the flow field[1]. Here we take this term to be a 5×5 rectangular region to match the size of the median filter in **Classic-C**. Figure 5 shows the neighborhood for the standard pairwise model and the non-local term.

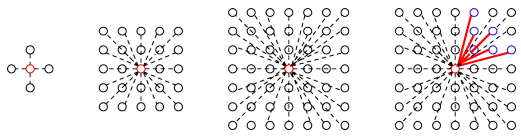


Fig. 5 From left to right, neighborhood structure for the center (red) pixel for the standard pairwise model, the unweighted non-local model, the unweighted non-local model with a larger neighborhood, and the weighted non-local model. The standard pairwise model connects a center pixel with its nearest neighbors, while the non-local term connects a pixel with many pixels in a large spatial neighborhood. By assigning larger weights (thicker red edges) to neighbors that are more likely to be on the same surface (blue circles), the weighted non-local model incorporates spatial scene structure information.

It is usually difficult to directly optimize the objective (2) with a large spatial term. A common practice is to relax the

[1] Bruhn *et al.* [13] also integrated information over a local region in a global method but did so for the data term.

objective with an auxiliary flow field as

$$\begin{aligned} E_A(\mathbf{u},\mathbf{v},\hat{\mathbf{u}},\hat{\mathbf{v}}) = & \qquad (3)\\ & \sum_{i,j}\Big\{\rho_D(I_1(i,j)-I_2(i+u_{i,j},j+v_{i,j}))\\ & \quad +\lambda[\rho_S(u_{i,j}-u_{i+1,j})+\rho_S(u_{i,j}-u_{i,j+1})\\ & \qquad +\rho_S(v_{i,j}-v_{i+1,j})+\rho_S(v_{i,j}-v_{i,j+1})]\Big\}\\ & +\lambda_C(||\mathbf{u}-\hat{\mathbf{u}}||^2+||\mathbf{v}-\hat{\mathbf{v}}||^2)\\ & +\lambda_N\sum_{i,j}\sum_{(i',j')\in\mathcal{N}_{i,j}}(|\hat{u}_{i,j}-\hat{u}_{i',j'}|+|\hat{v}_{i,j}-\hat{v}_{i',j'}|), \end{aligned}$$

where $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ denote an auxiliary flow field and λ_C is a scalar weight. A third (coupling) term encourages $\hat{\mathbf{u}},\hat{\mathbf{v}}$ and $\mathbf{u},\mathbf{v}$ to be the same (cf. [48,57]). Here the notation implies a pixelwise sum of squared errors between the auxiliary and main flow fields.

The connection to median filtering (as a denoising method) derives from the fact that there is a direct relationship between the median and L1 minimization. Consider a simplified version of Eq. (3) with just the coupling and non-local terms, where

$$E(\hat{\mathbf{u}})=\lambda_C||\mathbf{u}-\hat{\mathbf{u}}||^2+\lambda_N\sum_{i,j}\sum_{(i',j')\in\mathcal{N}_{i,j}}|\hat{u}_{i,j}-\hat{u}_{i',j'}|. \quad (4)$$

While minimizing this is similar to median filtering $\mathbf{u}$, there are two differences. First, the non-local term minimizes the L1 distance between the central value and all flow values in its neighborhood except itself. Second, Eq. (4) incorporates information about the data term through the coupling equation; median filtering the flow ignores the data term.

The formal connection between Eq. (4) and median filtering[2] is provided by Li and Osher [29] who show that minimizing Eq. (4) is related to a different median computation

$$\hat{u}_{i,j}^{(k+1)} = \text{median}(\text{Neighbors}^{(k)}\cup\text{Data}) \quad (5)$$

where $\text{Neighbors}^{(k)}=\{\hat{u}_{i',j'}^{(k)}\}$ for $(i',j')\in\mathcal{N}_{i,j}$ and $\hat{\mathbf{u}}^{(0)}=\mathbf{u}$ as well as

$$\text{Data}=\{u_{i,j},u_{i,j}\pm\tfrac{\lambda_N}{\lambda_C},u_{i,j}\pm\tfrac{2\lambda_N}{\lambda_C}\cdots,u_{i,j}\pm\tfrac{|\mathcal{N}_{i,j}|\lambda_N}{2\lambda_C}\},$$

where $|\mathcal{N}_{i,j}|$ denotes the (even) number of neighbors of (i,j). Note that the set of "data" values is balanced with an equal number of elements on either side of the value $u_{i,j}$ and that information about the data term is included through $u_{i,j}$. Repeated application of Eq. (5) converges rapidly [29].

Observe that, as λ_N/λ_C increases, the weighted data values on either side of $u_{i,j}$ move away from the values of Neighbors and cancel each other out. As this happens, Eq. (5) approximates the median at the first iteration

$$\hat{u}_{i,j}^{(1)}\approx\text{median}(\text{Neighbors}^{(0)}\cup\{u_{i,j}\}). \quad (6)$$

Eq. (3) thus combines the original objective with an approximation to the median, the influence of which is controlled by λ_N/λ_C. Note in practice the weight λ_C on the coupling term is usually small or is steadily increased from small values [49,57]. We optimize the new objective (3) by alternately minimizing

$$\begin{aligned} E_O(\mathbf{u},\mathbf{v}) = & \sum_{i,j}\Big\{\rho_D(I_1(i,j)-I_2(i+u_{i,j},j+v_{i,j}))\\ & \quad +\lambda[\rho_S(u_{i,j}-u_{i+1,j})+\rho_S(u_{i,j}-u_{i,j+1})\\ & \qquad +\rho_S(v_{i,j}-v_{i+1,j})+\rho_S(v_{i,j}-v_{i,j+1})]\Big\}\\ & +\lambda_C(||\mathbf{u}-\hat{\mathbf{u}}||^2+||\mathbf{v}-\hat{\mathbf{v}}||^2) \qquad (7) \end{aligned}$$

and

$$\begin{aligned} E_M(\hat{\mathbf{u}},\hat{\mathbf{v}}) = & \lambda_C(||\mathbf{u}-\hat{\mathbf{u}}||^2+||\mathbf{v}-\hat{\mathbf{v}}||^2) \qquad (8)\\ & +\lambda_N\sum_{i,j}\sum_{(i',j')\in\mathcal{N}_{i,j}}(|\hat{u}_{i,j}-\hat{u}_{i',j'}|+|\hat{v}_{i,j}-\hat{v}_{i',j'}|). \end{aligned}$$

We find that optimization of the coupled set of equations is superior in terms of EPE performance than optimization of the objective (2).

The alternating optimization strategy first holds $\hat{\mathbf{u}},\hat{\mathbf{v}}$ fixed and minimizes Eq. (7) w. r. t. $\mathbf{u},\mathbf{v}$. Then, with $\mathbf{u},\mathbf{v}$ fixed, we minimize Eq. (8) w. r. t. $\hat{\mathbf{u}},\hat{\mathbf{v}}$. Note that Eqs. (4) and (8) can be minimized by repeated application of Eq. (5); we use this approach with 5 iterations. We perform 10 steps of alternating optimizations at every pyramid level and change λ_C logarithmically from 10^{-4} to 10^2. During the first and second GNC stages, we set $\mathbf{u},\mathbf{v}$ to be $\hat{\mathbf{u}},\hat{\mathbf{v}}$ after every warping step (this replacement step helps reach solutions with lower energy and EPE than without performing this step; see **Classic-C-A-noRep** in Tables 6 and 7). In the end, we take $\hat{\mathbf{u}},\hat{\mathbf{v}}$ as the final flow field estimate. The other parameters are $\lambda=5,\lambda_N=1$.

Alternately optimizing this new objective function (**Classic-C-A**) leads to similar results as the baseline **Classic-C** (Table 6). We also compare the energy of these solutions using the new objective and find the alternating optimization produces the lowest energy solutions, as shown in Table 7.

We find that approximately optimizing the new objective by changing λ_C logarithmically from 10^{-4} to 10^{-1} has slightly better EPE results but higher energy solutions (**Classic-C-A-II**). We also try replacing the absolute value by the Charbonnier penalty function and using the conjugate gradient descent method [3] to solve Eq. (4) but obtain results with slightly worse EPE performance and higher energy.

[2] Hsiao *et al.* [23] established the connection in a slightly different way.

	Avg. EPE	significance	p-value
Classic-C	0.298	—	—
Classic-C-A	0.305	0	0.8125
Classic-C-A-noRep	0.309	0	0.5781
Classic-C-A-II	0.296	0	0.7188
Classic-C-A-CGD	0.305	0	0.5625

Table 6 Average end-point error (EPE) on the Middlebury *training set* is shown for the new model with alternating optimization (**Classic-C-A**). Please refer to Table 15 for the detailed EPE results on each training sequence.

In summary, we show that the heuristic median filtering step in **Classic-C** can now be viewed as energy minimization of a new objective with a non-local term. The explicit formulation emphasizes the value of robustly integrating information over large neighborhoods and enables the improved model described below.

6 Improved Model

By formalizing the median filtering heuristic as an explicit objective function, we can find ways to improve it. While median filtering in a large neighborhood has advantages as we have seen, it also has problems. A neighborhood centered on a corner or thin structure is dominated by the surround and computing the median results in oversmoothing as illustrated in Fig. 1.

Examining the non-local term suggests a solution. For a given pixel, if we know which other pixels in the area belong to the same surface, we can weight them more highly. The modification to the objective function is achieved by introducing a weight into the non-local term [14, 19]:

$$\sum_{i,j} \sum_{(i',j')\in N_{i,j}} w_{i,j}^{i',j'} \left(|\hat{u}_{i,j} - \hat{u}_{i',j'}| + |\hat{v}_{i,j} - \hat{v}_{i',j'}|\right), \quad (9)$$

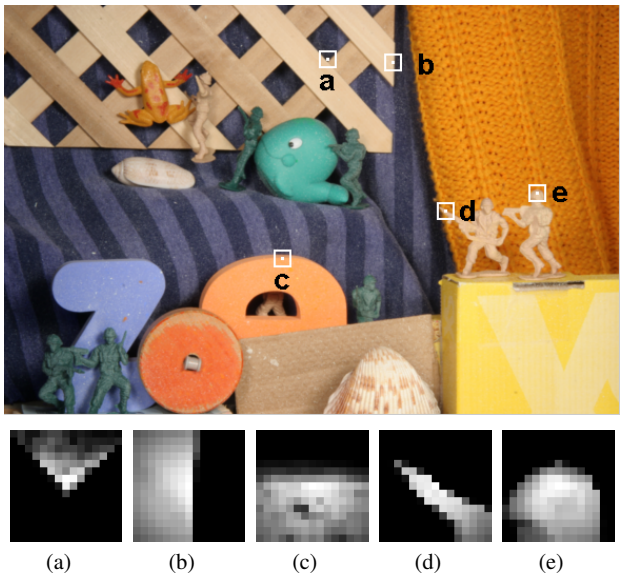
where $w_{i,j}^{i',j'}$ represents how likely pixel i', j' is to belong to the same surface as i, j.

Of course, we do not know $w_{i,j}^{i',j'}$, but can approximate it. We draw ideas from [39, 53, 56] to define the weights according to their spatial distance, their color-value distance, and their occlusion state as

$$w_{i,j}^{i',j'} \propto \exp\left\{-\frac{|i-i'|^2+|j-j'|^2}{2\sigma_1^2} - \frac{|\mathbf{I}(i,j)-\mathbf{I}(i',j')|^2}{2\sigma_2^2 n_c}\right\} \frac{o(i',j')}{o(i,j)}, \quad (10)$$

where $\mathbf{I}(i,j)$ is the color vector in the Lab space, n_c is the number of color channels, $\sigma_1 = 7, \sigma_2 = 7$, and the occlusion variable $o(i,j)$ is calculated using Eq. (22) in [39] as

$$o(i,j) = \exp\left\{-\frac{d^2(i,j)}{2\sigma_d^2} - \frac{\left(I(i,j) - I(i+u_{i,j}, j+v_{i,j})\right)^2}{2\sigma_e^2}\right\}, \quad (11)$$



(a) (b) (c) (d) (e)

Fig. 6 Neighbor weights of the proposed weighted non-local term at different positions in the "Army" sequence. We use color, spatial distance, and occlusion cues to determine whether the neighboring pixels are likely to belong to the same surface. Among these cues, color is the most powerful.

where $d(i,j)$ is the one-sided divergence function, defined as

$$d(i,j) = \begin{cases} \mathrm{div}(i,j), & \mathrm{div}(i,j) < 0 \\ 0, & \text{otherwise} \end{cases} \quad (12)$$

in which the flow divergence $\mathrm{div}(i,j)$ is

$$\mathrm{div}(i,j) = \frac{\partial}{\partial x} u(i,j) + \frac{\partial}{\partial y} v(i,j), \quad (13)$$

where $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ are respectively the horizontal and vertical flow derivatives. The occlusion variable $o(i,j)$ is near zero for occluded pixels and near one for non-occluded pixels. We set the parameters in Eq. (11) as $\sigma_d = 0.3$ and $\sigma_e = 20$; this is the same as in [39].

Examples of such weights are shown for several 15×15 neighborhoods in Figure 6; bright values indicate higher weights. Note the neighborhood labeled **d**, corresponding to the rifle. Since pixels on the rifle are in the minority, an unweighted median oversmooths (**Classic++** in Fig. 1). The weighted term instead robustly estimates the motion using values on the rifle. A closely related piece of work is [37], which uses the intervening contour to define affinities among neighboring pixels for the local Lucas and Kanade [31] method. However it only uses this scheme to estimate motion for sparse points and then interpolates the dense flow field.

We approximately solve for $\hat{\mathbf{u}}$ (and similarly $\hat{\mathbf{v}}$) using the following weighted median problem

$$\min_{\hat{u}_{i,j}} \sum_{(i',j')\in N_{i,j}\cup\{i,j\}} w_{i,j}^{i',j'} |\hat{u}_{i,j} - u_{i',j'}|, \quad (14)$$

	Sum	Venus	Dimetrodon	Hydrangea	RubberWhale	Grove2	Grove3	Urban2	Urban3
Classic-C	13.013	0.817	0.903	1.202	0.674	2.166	3.144	1.954	2.153
Classic-C w/o MF	14.629	0.886	0.945	1.299	0.725	2.315	3.513	2.234	2.712
Classic-C-A	12.489	0.784	0.889	1.139	0.666	2.064	2.976	1.922	2.049
Classic-C-A-noRep	13.076	0.790	0.894	1.165	0.670	2.092	3.143	2.005	2.317
Classic-C-A-II	13.308	0.830	0.915	1.235	0.686	2.223	3.247	1.990	2.182
Classic-C-A-CGD	13.466	0.833	0.909	1.224	0.674	2.213	3.357	2.020	2.236

Table 7 Eq. (3) energy ($\times 10^6$) for the computed flow fields on the Middlebury *training set*. The alternating optimization strategy (**Classic-C-A**) produces the lower energy solutions than the median filtering heuristic.

	Avg. EPE	significance	p-value
Classic+NL	0.221	—	—
Classic+NL-Full	0.222	0	0.8203
Classic+NL-Fast	0.221	0	0.3125
RGB	0.240	**1**	**0.0156**
HSV	0.231	**1**	**0.0312**
LUV	0.226	0	0.5625
Gray	0.253	**1**	**0.0078**
w/o color	0.283	**1**	**0.0156**
w/o occ	0.226	0	0.1250
w/o spa	0.223	0	0.5625
$\sigma_2 = 5$	0.221	0	1.0000
$\sigma_2 = 10$	0.224	0	0.2500
$\lambda = 1$	0.236	0	0.1406
$\lambda = 9$	0.244	0	0.1016
11×11	0.223	0	0.5938
19×19	0.220	0	0.8750

Table 8 Average end-point error (EPE) on the Middlebury *training set* is shown for the the improved model and its variants. Please refer to Table 16 for the detailed results.

using the formula (3.13) in [29] for all the pixels (**Classic+NL-Full**). Note if all the weights are equal, the solution is just the median. In practice, we can adopt a fast version (**Classic+NL**) without performance loss: Given a current estimate of the flow, we detect motion boundaries using a Sobel edge detector and dilate these edges with a 5×5 mask to obtain flow boundary regions. In these regions we use the weighting in Eq. (10) in a 15×15 neighborhood. In the non-boundary regions, we use equal weights in a 5×5 neighborhood to compute the median.

To further reduce the computation, we can adopt a two-stage GNC process and perform 3 warping steps per pyramid level. This fast version (**Classic+NL-Fast**) has nearly the same overall performance, with a slight decline in performance on the "Urban3" sequence, which has large motions; with an iterative warping scheme, large motions require more iterations.

Tables 8 and 9 show that the weighted non-local term (**Classic+NL**) improves the accuracy on both the training and the test sets, especially in the motion boundary regions. Note that the fine detail of the "rifle" is preserved in Figure 1(e). At the writing of this paper (Sep. 2012), **Classic+NL** ranks 13$^{\text{th}}$ in both AAE and EPE. Figures 7 and 8 show some of the results on the Middlebury dataset.

Computational time. The running time on the test "Urban" sequence is about 1.5 minutes for **HS**, 6 minutes for **Classic++**, about 8 minutes for **Classic+NL**, about 26 minutes for **Classic+NL-Full**, and about 1.6 minutes for **Classic+NL-Fast** in MATLAB on a 64-bit Linux desktop with 8G of memory. The additional cost from **HS** to **Classic++** comes from the GNC stage and the non-convex penalty function. The additional cost from **Classic++** to **Classic+NL** comes from the weighted median filtering step for detected motion boundaries. Applying the weighted median operation on all the pixels (**Classic+NL-Full**) increases the running time by more than three times with little performance gain. Using fewer iterations (**Classic+NL-Fast**) can significantly reduce the computational cost with little performance loss, especially on sequences with small motion. Note that we solve the weighted median problem at each pixel individually and do not reuse the sorting results from neighboring pixels. Future work should consider reformulating the weighted median filtering so that a convolution-type operation can be used to reduce the computational costs.

	Avg. Rank	**Avg. EPE**	**Avg. EPE** near boundary
Classic++	32.7	0.406	0.980
Classic++Gradient	33.5	0.430	1.042
Classic+NL	17.2	0.319	0.689
Classic+NL-Full	17.5	0.316	0.676

Table 9 Average end-point error (EPE) on the Middlebury *test set* for the **Classic++** model with two different preprocessing techniques and its improved model. Please refer to Table 11 for the detailed EPE results, Figs. 11 and 12 for the screen shots of the Middlebury public table.

We study some variants of the weighted non-local term (**Classic+NL**). Table 8 shows the importance of each term in determining the weight and influence of the parameter setting on the final results. Using different color spaces results in some performance decline. Using grayscale pixel values (**Gray**) or not using the static image information (**w/o color**) results in significant degradation in performance. Without occlusion (**w/o occ**) or spatial distance (**w/o spa**) cues does not degrade the performance significantly. The method is robust to the setting of σ_2 for the color cue and 5 and 10 perform similarly as the default 7. The default λ is 3, while 1 and 9 result in some loss in performance. We also study the maximum size of the neighborhood for the non-local term and find 11×11 gives similar performance while 19×19 is slightly better.

Results on the MIT dataset.

To test the robustness of these models on other data, we applied **HS**, **Classic-C**, and **Classic+NL** to sequences from

(a) "Old" **HS** [42] (b) "New" **HS** (c) **Classic++** (d) **Classic+NL** (e) Ground truth (f) First frame

Fig. 7 Results on the Middlebury test set. Top to bottom: "Teddy", "Wooden", and "Grove". **Classi+NL** uses information from the color image to detect and preserve fine motion details. Note that the ground truth from the Middlebury website has been compressed and has low quality than the actual one.

the MIT dataset [30], and compared the estimated flow fields to the human labeled ground truth. Note only five of the eight test sequences in [30] are available on-line; these are tested here.

Fig. 9 and Table 10 show the results on these sequences, which are very different in nature from the Middlebury set and include an outdoor scene as well as a scene of a fish tank. The results are compared with the CLG method [13] used in [30]. It is important to point out that the CLG method was tuned to obtain the optimal results on the test sequences. Our method had no such tuning and we used the same parameters as those used in all the other experiments. This suggests that training on the Middlebury data results in a method that generalizes to other sequences. The only place where this fails is on the "fish" sequence where there is transparent motion in a liquid medium; the statistics in this sequence are very different from the Middlebury training data.

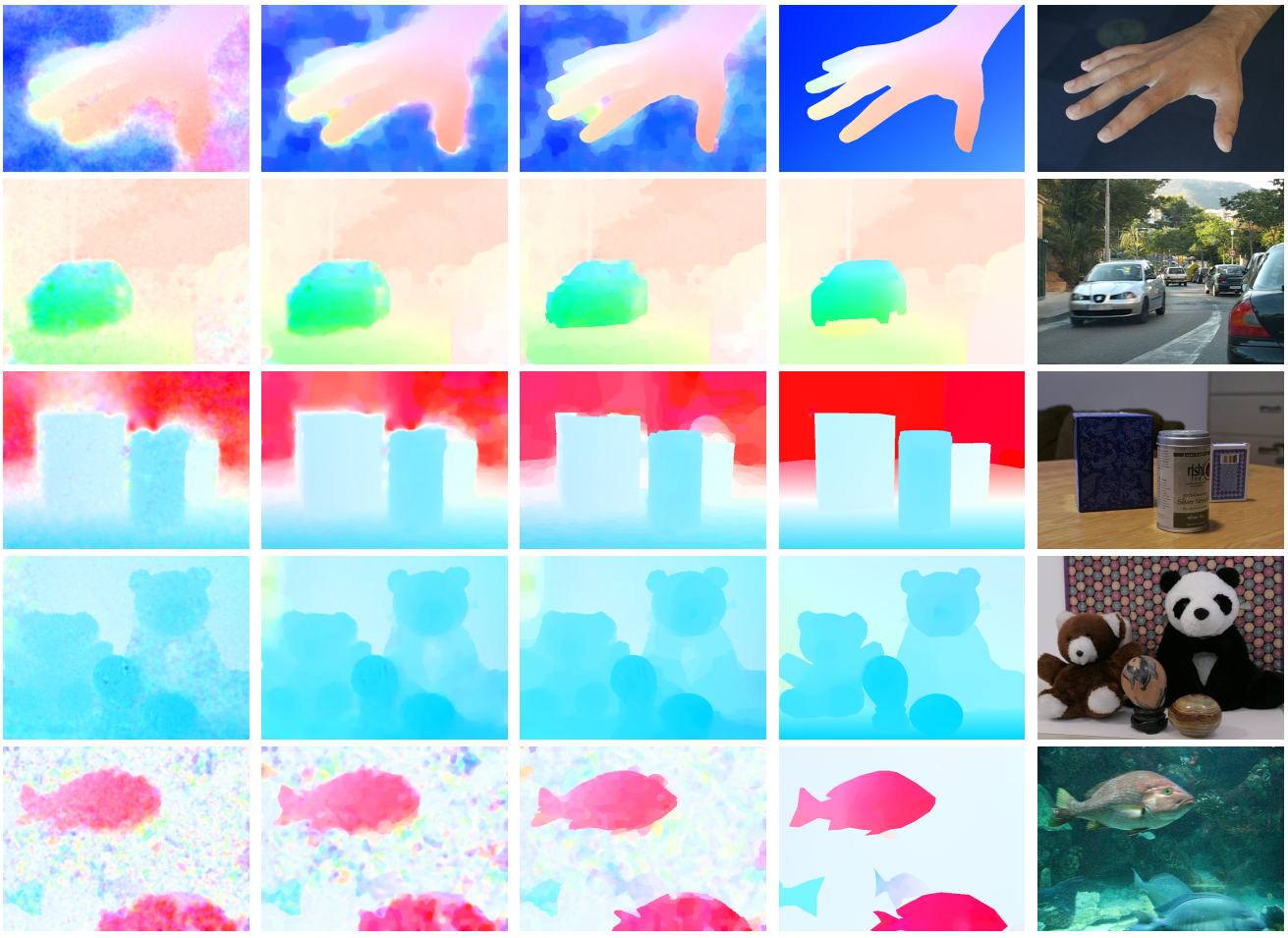
	Average	Table	Hand	Toy	Fish	CameraMotion
CLG [13,30]	1.239	0.976	4.181	0.456	0.196	0.385
HS	2.129	1.740	6.108	0.620	1.309	0.869
Classic-C	1.345	1.064	3.428	0.482	1.061	0.690
Classic+NL	1.106	0.91	2.75	0.487	0.772	0.611

Table 10 Results on the MIT dataset [30]. Average end-point error (EPE). The CLG [13] method was tuned for each sequence [30].

Closely-related work: Werlberger *et al.* [50] independently propose a non-local term for optical flow estimation and the spatial term is similar to our non-local term. They use normalized cross correlation as the data term to deal with lighting changes and optimize their objective function by a primal-dual method. Their work is motivated by the success of the non-local regularization [14] in image restoration and stereo. Our work is inspired by the success for the heuristic median filtering step in flow estimation and we formalize

(d) **Classic+NL** (e) Ground truth (e) First frame (d) **Classic+NL** (e) Ground truth (f) First frame

Fig. 8 Results on other Middlebury test sequences.

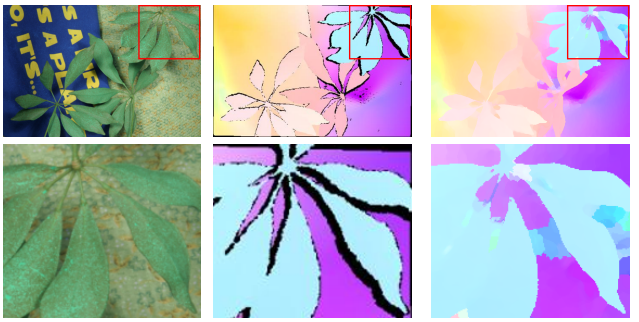


(a) **HS** (b) **Classic-C** (c) **Classic+NL** (d) Ground truth (e) First frame

Fig. 9 Results on MIT sequences.

the median filtering heuristic as adopting a non-local regularization term.

Limitations: Classic+NL produces larger errors in occlusion regions on some sequences, such as "Schefflera" shown in Fig. 10. The classical flow formulation assumes that every pixel at the current frame has a corresponding pixel at the next frame. However this assumption breaks down in regions of occlusion. Pixels that are occluded by some foreground objects in one frame do not have corresponding pixels in the next, resulting in large errors with classical formulations. In contrast, a layered model [46] may provide a principled way to reason about occlusions. The motion model developed in this paper has enabled the a recent layered approach [43] to achieve a consistent improvement over the **Classic+NL**



(a) First frame (b) Ground truth (c) Estimated flow field

Fig. 10 Occlusions are not modeled by **Classic+NL** and may cause problems in the estimated flow field. Dark pixels in the ground truth indicate occlusions.

method, in particular near occlusion and motion boundary regions.

7 Conclusions

When implemented using modern practices, classical optical flow formulations produce competitive results on the Middlebury training and test sets. To understand the "secrets" that help such basic formulations work well, we quantitatively studied various aspects of flow approaches from the literature, including their implementation details. Among the best practices, we found that using median filtering to denoise the flow after every warping step is key to improving accuracy, but that this increases the energy of the final result. Exploiting connections between median filtering and L1-based denoising, we showed that algorithms relying on a median filtering step are approximately optimizing a different objective that regularizes flow over a large spatial neighborhood. Understanding this enables us to design and optimize improved models that weight the neighbors adaptively in an extended image region. The MATLAB code is publicly available [1].

References

1. http://www.cs.brown.edu/people/dqsun.
2. http://gpu4vision.icg.tugraz.at.
3. http://www.gaussianprocess.org/gpml/code/matlab/util/minimize.m.
4. E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. Pyramid methods in image processing. *RCA Engineer*, 29(6):33–41, November 1984.
5. A. Bab-Hadiashar and D. Suter. Robust optic flow computation. *International Journal of Computer Vision*, 29(1):59–77, August 1998.
6. S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *IEEE International Conference on Computer Vision*, pages 1–8, 2007.
7. J. Barron, D. Fleet, and S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, February 1994.
8. J. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *European Conference on Computer Vision*, volume 588, pages 237–252, 1992.
9. M. Black and A. Jepson. Estimating optical-flow in segmented images using variable-order parametric models with local deformations. *IEEE Transaction on Pattern Analysis Machine Intelligence*, 18(10):972–986, October 1996.
10. M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63:75–104, 1996.
11. A. Blake and A. Zisserman. *Visual Reconstruction*. The MIT Press, Cambridge, Massachusetts, 1987.
12. T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision*, pages 25–36, 2004.
13. A. Bruhn, J. Weickert, and C. Schnörr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, February 2005.
14. A. Buades, B. Coll, and J. Morel. A non-local algorithm for image denoising. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 60–65, 2005.
15. P. J. Burt, C. Yen, and X. Xu. Local correlation measures for motion analysis: A comparative study. *Processings of IEEE Pattern Recognition and Image Processing*, pages 269–274, 1982.
16. Z. Chen, Y. Wu, and J. Wang. Decomposing and regularizing sparse/non-sparse components for motion field estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1176–1183, 2012.
17. A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, Providence, USA, 2012.
18. S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transaction on Pattern Analysis Machine Intelligence*, 6(6):721–741, November 1984.
19. G. Gilboa and S. Osher. Nonlocal operators with applications to image processing. *SIAM Multiscale Modeling and Simulation*, 7:1005–1028, 2008.
20. F. Glaer, G. Reynolds, and P. Anandan. Scene matching by hierarchical correlation. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 432–441, 1983.
21. B. Horn. *Robot Vision*. MIT Press, 1986.
22. B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 16(1-3):185–203, August 1981.
23. I. Hsiao, A. Rangarajan, and G. Gindi. A new convex edge-preserving median prior with applications to tomography. *IEEE Transactions on Medical Imaging*, 22(5):580–585, May 2003.
24. K. Jia, X. Wang, and X. Tang. Optical flow estimation using learned sparse model. In *IEEE International Conference on Computer Vision*, pages 2391–2398, 2011.
25. R. G. Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 29(6):1153–1160, December 1981.
26. P. Krähenbühl and V. Koltun. Efficient nonlocal regularization for optical flow. In *European Conference on Computer Vision*, 2012.
27. C. Lei and Y.-H. Yang. Optical flow estimation on coarse-to-fine region-trees using discrete optimization. In *IEEE International Conference on Computer Vision*, pages 1562–1569, 2009.
28. V. Lempitsky, S. Roth, and C. Rother. FusionFlow: Discrete-continuous optimization for optical flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.
29. Y. Li and S. Osher. A new median formula with applications to PDE based denoising. *Communications in Mathematical Sciences*, 7(3):741–753, September 2009.
30. C. Liu, W. T. Freeman, E. H. Adelson, and Y. Weiss. Human-assisted motion annotation. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

Table 11 Models. Average end-point error (EPE) on the Middlebury optical flow benchmark (*test set*). The ranking information was at the writing of the paper (Sep. 2012).

	Rank	**Average**	Army	Mequon	Schefflera	Wooden	Grove	Urban	Yosemite	Teddy
HS	49.0	0.501	0.12	0.25	0.45	0.24	0.95	0.83	0.24	0.93
Classic-C	34.8	0.408	0.10	0.23	0.45	0.20	0.88	0.47	0.16	0.77
Classic-L	42.7	0.530	0.10	0.24	0.47	0.21	0.92	1.23	0.20	0.87
HS-brightness	N/A	0.759	0.21	0.89	1.13	0.42	0.93	0.70	0.18	1.61
Classic-C-brightness	N/A	0.726	0.39	0.95	1.12	0.42	0.87	0.48	0.13	1.45
Classic-L-brightness	N/A	0.603	0.17	0.64	0.84	0.32	0.90	0.48	0.13	1.34
HS [42]	66.2	0.872	0.22	0.61	1.01	0.78	1.26	1.43	0.16	1.51
BA (Classic-L) [42]	59.6	0.746	0.18	0.58	0.95	0.49	1.08	1.43	0.15	1.11
Adaptive [48]	28.5	0.401	0.09	0.23	0.54	0.18	0.88	0.50	0.14	0.65
Complementary OF [59]	31.6	0.485	0.10	0.20	0.35	0.19	0.87	1.46	0.11	0.60
Classic++	32.7	0.406	0.09	0.23	0.43	0.20	0.87	0.47	0.17	0.79
Classic++Gradient	33.5	0.430	0.08	0.17	0.49	0.21	0.94	0.55	0.17	0.83
Classic+NL	17.2	0.319	0.08	0.22	0.29	0.15	0.64	0.52	0.16	0.49
Classic+NL-Full	17.5	0.316	0.08	0.24	0.28	0.15	0.63	0.49	0.16	0.50

Table 12 Models and pre-processing. Average end-point error (EPE) on the Middlebury *training set* for the classical model and different penalty functions. By default, the input sequences were preprocessed using ROF texture decomposition; "brightness" means no preprocessing is performed. The statistical significance is tested using the Wilcoxon signed rank test between each method and the baseline (**Classic-C**).

	Average	Venus	Dimetrodon	Hydrangea	RubberWhale	Grove2	Grove3	Urban2	Urban3	signif.	p-value
Classic-C	0.298	0.281	0.152	0.165	0.093	0.158	0.627	0.348	0.562	—	—
Classic-C-brightness	0.288	0.268	0.166	0.215	0.134	0.146	0.584	0.352	0.437	0	0.9453
HS	0.384	0.337	0.219	0.189	0.118	0.204	0.688	0.463	0.853	**1**	**0.0078**
HS-brightness	0.387	0.335	0.226	0.252	0.154	0.185	0.639	0.564	0.743	**1**	**0.0078**
Classic-L	0.319	0.294	0.193	0.175	0.095	0.166	0.648	0.374	0.604	**1**	**0.0078**
Classic-L-brightness	0.325	0.292	0.207	0.274	0.145	0.158	0.588	0.451	0.484	0	0.2969

Table 13 Pre-Processing. Average end-point error (EPE) on the Middlebury *training set* for the baseline method (**Classic-C**) using different pre-processing techniques. The regularization weight λ parameter was tuned for each method to achieve optimal performance. The statistical significance is tested using the Wilcoxon signed rank test between each method and the baseline (**Classic-C**).

	Average	Venus	Dimetrodon	Hydrangea	RubberWhale	Grove2	Grove3	Urban2	Urban3	signif.	p-value
Classic-C	0.298	0.281	0.152	0.165	0.093	0.158	0.627	0.348	0.562	—	—
Gradient	0.305	0.288	0.141	0.167	0.092	0.165	0.614	0.385	0.588	0	0.4609
Gaussian	0.281	0.268	0.146	0.226	0.141	0.137	0.582	0.335	0.413	0	0.5469
Gaussian + Dx + Dy	0.290	0.280	0.126	0.174	0.105	0.154	0.588	0.470	0.420	0	0.6406
Dx + Dy	0.301	0.286	0.122	0.166	0.099	0.161	0.616	0.443	0.518	0	1.0000
Sobel edge[45]	0.417	0.334	0.149	0.184	0.130	0.194	0.757	0.451	1.135	**1**	**0.0156**
Laplacian [28]	0.430	0.374	0.170	0.176	0.096	0.175	0.756	0.464	1.232	**1**	**0.0078**
Laplacian 1:1	0.301	0.296	0.179	0.193	0.109	0.157	0.606	0.349	0.520	0	0.6641
Texture 4:1	0.286	0.271	0.159	0.175	0.100	0.154	0.587	0.349	0.490	0	0.5312
Unnormalized texture	0.298	0.279	0.152	0.166	0.092	0.158	0.623	0.348	0.563	0	0.3750

Table 14 Model and Methods. Average end-point error (EPE) on the Middlebury *training set* for the baseline model (**Classic-C**) using different algorithm and modeling choices. The statistical significance is tested using the Wilcoxon signed rank test between each method and the baseline (**Classic-C**).

	Average	Venus	Dimetrodon	Hydrangea	RubberWhale	Grove2	Grove3	Urban2	Urban3	signif.	p-value
Classic-C	0.298	0.281	0.152	0.165	0.093	0.158	0.627	0.348	0.562	—	—
3 **warping steps**	0.304	0.283	0.122	0.163	0.095	0.150	0.622	0.357	0.644	0	0.9688
Down-0.5	0.298	0.280	0.152	0.166	0.092	0.158	0.626	0.349	0.562	0	1.0000
Down-0.95	0.298	0.281	0.151	0.168	0.099	0.165	0.661	0.339	0.523	0	0.9375
w/o GNC	0.354	0.303	0.160	0.171	0.105	0.183	0.835	0.316	0.759	0	0.1094
Bilinear	0.302	0.284	0.144	0.167	0.099	0.160	0.637	0.363	0.563	0	0.1016
w/o TAVG	0.306	0.288	0.149	0.167	0.093	0.163	0.647	0.345	0.593	0	0.1562
Central	0.300	0.272	0.156	0.169	0.092	0.159	0.608	0.349	0.597	0	0.7266
7-point [13]	0.302	0.282	0.168	0.171	0.091	0.163	0.601	0.360	0.584	0	0.3125
Deriv-warp	0.297	0.283	0.153	0.165	0.092	0.159	0.636	0.333	0.552	0	0.9531
Bicubic-II	0.290	0.276	0.132	0.152	0.083	0.142	0.624	0.338	0.571	**1**	**0.0391**
Deriv-warp-II	0.287	0.264	0.155	0.152	0.085	0.145	0.616	0.333	0.546	**1**	**0.0156**
Warp-deriv-II	0.288	0.267	0.155	0.151	0.085	0.147	0.630	0.328	0.542	**1**	**0.0391**
C-L ($\lambda = 0.6$)	0.303	0.290	0.158	0.171	0.094	0.158	0.611	0.367	0.579	0	0.1562
L-C ($\lambda = 2$)	0.306	0.281	0.174	0.173	0.096	0.164	0.662	0.343	0.557	0	0.1562
GC-0.45 ($\lambda = 3$)	0.292	0.280	0.145	0.165	0.092	0.154	0.612	0.340	0.546	**1**	**0.0156**
GC-0.25 ($\lambda = 0.7$)	0.298	0.283	0.128	0.169	0.094	0.150	0.617	0.353	0.594	0	1.0000
MF 3×3	0.305	0.287	0.155	0.168	0.094	0.162	0.616	0.372	0.583	0	0.1016
MF 7×7	0.305	0.281	0.152	0.173	0.095	0.174	0.676	0.330	0.557	0	0.5625
$2\times$ **MF**	0.300	0.279	0.152	0.167	0.093	0.163	0.650	0.339	0.555	0	1.0000
$5\times$ **MF**	0.305	0.278	0.152	0.171	0.093	0.172	0.682	0.329	0.561	0	0.6875
w/o MF	0.352	0.307	0.168	0.199	0.113	0.217	0.705	0.423	0.684	**1**	**0.0078**
Classic++	0.285	0.271	0.128	0.153	0.081	0.139	0.614	0.336	0.555	**1**	**0.0078**

Average endpoint error	avg. rank	Army (Hidden texture) GT	Army (Hidden texture) im0	Army (Hidden texture) im1	Mequon (Hidden texture) GT	Mequon (Hidden texture) im0	Mequon (Hidden texture) im1	Schefflera (Hidden texture) GT	Schefflera (Hidden texture) im0	Schefflera (Hidden texture) im1	Wooden (Hidden texture) GT	Wooden (Hidden texture) im0	Wooden (Hidden texture) im1	Grove (Synthetic) GT	Grove (Synthetic) im0	Grove (Synthetic) im1	Urban (Synthetic) GT	Urban (Synthetic) im0	Urban (Synthetic) im1	Yosemite (Synthetic) GT	Yosemite (Synthetic) im0	Yosemite (Synthetic) im1	Teddy (Stereo) GT	Teddy (Stereo) im0	Teddy (Stereo) 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
MDP-Flow2 [74]	4.2	0.08 4	0.21 2	0.07 8	**0.15** 1	**0.48** 1	**0.11** 1	**0.20** 1	**0.40** 1	**0.14** 1	0.15 8	0.80 14	0.08 6	0.63 6	0.93 6	0.43 6	**0.26** 1	**0.76** 1	0.23 3	0.11 9	0.12 6	0.17 11	**0.38** 1	**0.79** 1	**0.44** 1
ADF [71]	9.0	0.08 4	0.22 6	**0.06** 1	0.18 3	0.62 8	0.14 8	0.29 14	0.71 17	0.17 8	0.16 16	0.91 26	**0.07** 1	0.69 12	1.03 12	0.47 8	0.43 11	0.91 4	0.28 6	0.12 15	0.12 6	0.20 19	0.43 2	0.88 2	0.63 6
IROF++ [61]	9.2	0.08 4	0.23 8	0.07 8	0.21 17	0.68 15	0.17 18	0.28 10	0.63 8	0.19 18	0.15 8	0.73 10	0.09 13	0.60 3	0.89 4	0.42 5	0.43 11	1.08 10	0.31 12	0.10 4	0.12 6	0.12 4	0.47 6	0.98 8	0.68 11
Layers++ [38]	9.5	0.08 4	0.21 2	0.07 8	0.19 7	0.56 3	0.17 18	**0.20** 1	**0.40** 1	0.18 12	**0.13** 1	0.58 2	**0.07** 1	**0.48** 1	**0.70** 1	0.33 2	0.47 20	1.01 6	0.33 18	0.15 38	0.14 31	0.24 32	0.46 4	0.88 2	0.72 14
nLayers [60]	10.2	**0.07** 1	**0.19** 1	**0.06** 1	0.22 20	0.59 4	0.19 29	0.25 6	0.54 5	0.20 27	0.15 8	0.84 19	0.08 6	0.53 2	0.78 2	0.34 3	0.44 15	0.84 2	0.30 10	0.13 23	0.13 19	0.20 19	0.47 6	0.97 7	0.67 9
Sparse-NonSparse [58]	13.7	0.08 4	0.23 8	0.07 8	0.22 20	0.73 22	0.18 21	0.28 10	0.64 9	0.19 18	0.14 4	0.71 7	0.08 6	0.67 11	0.99 11	0.48 11	0.49 24	1.06 8	0.32 15	0.14 30	**0.11** 1	0.28 45	0.49 10	0.98 8	0.73 17
ALD-Flow [72]	13.8	**0.07** 1	0.21 2	**0.06** 1	0.19 7	0.64 13	0.13 4	0.30 17	0.73 18	0.15 3	0.17 22	0.92 30	**0.07** 1	0.78 21	1.14 21	0.59 21	0.33 3	1.30 22	**0.21** 1	0.12 15	0.12 6	0.28 45	0.54 18	1.19 21	0.73 17
COFM [62]	13.9	0.08 4	0.26 24	**0.06** 1	0.18 3	0.62 8	0.14 8	0.30 17	0.74 19	0.19 18	0.15 8	0.86 21	**0.07** 1	0.79 22	1.14 21	0.74 36	0.35 6	0.87 3	0.28 6	0.14 30	0.12 6	0.28 45	0.49 10	0.94 5	0.71 13
TC-Flow [47]	14.2	**0.07** 1	0.21 2	**0.06** 1	**0.15** 1	0.59 4	**0.11** 1	0.31 20	0.78 23	**0.14** 1	0.16 16	0.86 21	0.08 6	0.75 18	1.11 19	0.54 16	0.42 10	1.40 28	0.25 4	0.11 9	0.12 6	0.29 50	0.62 24	1.35 25	0.93 36
Efficient-NL [64]	14.2	0.08 4	0.22 6	**0.06** 1	0.23 30	0.73 22	0.18 21	0.32 22	0.75 20	0.18 12	0.14 4	0.72 9	0.08 6	0.60 3	0.88 3	0.43 6	0.57 33	1.11 14	0.35 24	0.14 30	0.13 19	0.25 35	0.48 8	0.90 4	0.63 6
LSM [40]	14.8	0.08 4	0.23 8	0.07 8	0.22 20	0.73 22	0.18 21	0.28 10	0.64 9	0.19 18	0.14 4	0.70 5	0.09 13	0.66 8	0.97 8	0.48 11	0.50 25	1.06 8	0.33 18	0.15 38	0.12 6	0.29 50	0.50 13	0.99 11	0.73 17
Ramp [66]	15.0	0.08 4	0.24 14	0.07 8	0.21 17	0.72 20	0.18 21	0.27 7	0.62 7	0.19 18	0.15 8	0.71 7	0.09 13	0.66 8	0.97 8	0.49 13	0.51 27	1.09 11	0.34 23	0.15 38	0.12 6	0.30 53	0.48 8	0.96 6	0.72 14
Classic+NL [31]	16.6	0.08 4	0.23 8	0.07 8	0.22 20	0.74 26	0.18 21	0.29 14	0.65 14	0.19 18	0.15 8	0.73 10	0.09 13	0.64 7	0.93 6	0.47 8	0.52 28	1.12 15	0.33 18	0.16 45	0.13 19	0.29 50	0.49 10	0.98 8	0.74 21
TV-L1-MCT [68]	16.8	0.08 4	0.23 8	0.07 8	0.24 33	0.77 28	0.19 29	0.32 22	0.76 22	0.19 18	0.14 4	0.69 4	0.09 13	0.72 14	1.03 12	0.60 22	0.54 31	1.10 13	0.35 24	0.11 9	0.12 6	0.20 19	0.54 18	1.04 15	0.84 28
Direct ZNCC [70]	17.5	0.09 16	0.25 18	0.07 8	0.19 7	0.70 18	0.13 4	0.43 30	1.00 33	0.15 3	**0.13** 1	**0.55** 1	0.08 6	0.86 29	1.23 27	0.73 32	0.53 30	1.22 17	0.38 33	0.14 30	0.13 19	0.27 42	0.44 3	0.99 11	**0.44** 1
IROF-TV [55]	18.4	0.09 16	0.25 18	0.08 22	0.22 20	0.77 28	0.19 29	0.30 17	0.70 16	0.19 18	0.18 25	0.93 32	0.11 28	0.73 15	1.04 14	0.56 19	0.44 15	1.69 43	0.31 12	0.09 3	**0.11** 1	0.12 4	0.50 13	1.08 17	0.73 17
MDP-Flow [26]	19.4	0.09 16	0.25 18	0.08 22	0.19 7	0.54 2	0.18 21	0.24 4	0.55 6	0.20 27	0.16 16	0.91 26	0.09 13	0.74 16	1.06 16	0.61 24	0.46 19	1.02 7	0.35 24	0.12 15	0.14 31	0.17 11	0.78 41	1.68 44	0.97 40
OF-Mol [48]	21.3	0.08 4	0.23 8	0.07 8	0.28 42	0.99 45	0.20 36	0.28 10	0.64 9	0.19 18	0.16 16	0.80 14	0.09 13	0.75 18	1.12 20	0.50 14	0.52 28	1.09 11	0.33 18	0.16 45	0.13 19	0.30 53	0.56 21	1.08 17	0.78 23
Sparse Occlusion [56]	21.3	0.09 16	0.24 14	0.08 22	0.22 20	0.63 10	0.19 29	0.38 28	0.91 27	0.18 12	0.17 22	0.85 20	0.09 13	0.75 18	1.09 18	0.47 8	0.34 4	1.00 5	0.26 5	0.22 63	0.22 68	0.28 45	0.53 17	1.13 19	0.67 9
OFH [39]	21.4	0.10 25	0.25 18	0.09 31	0.19 7	0.69 16	0.14 8	0.43 30	1.02 35	0.17 8	0.17 22	1.08 38	0.08 6	0.87 30	1.25 29	0.73 32	0.43 11	1.69 43	0.32 15	0.10 4	0.13 19	0.18 14	0.59 23	1.40 28	0.74 21
TrajectoryFlow [59]	22.2	0.10 25	0.26 24	0.07 8	0.20 14	0.73 22	0.13 4	0.37 27	0.94 28	0.15 3	**0.13** 1	0.67 3	**0.07** 1	0.82 24	1.23 27	0.54 16	0.66 36	1.42 31	0.44 34	0.16 45	0.14 31	0.37 61	0.58 22	1.25 22	0.79 24
NL-TV-NCC [25]	22.4	0.10 25	0.26 24	0.08 22	0.22 20	0.72 20	0.15 12	0.35 25	0.85 25	0.16 6	0.15 8	0.70 5	0.09 13	0.79 22	1.16 24	0.51 15	0.78 41	1.38 25	0.48 39	0.16 45	0.15 41	0.26 37	0.55 20	1.16 20	0.55 4
CostFilter [41]	22.5	0.10 25	0.27 30	0.08 22	0.20 14	0.63 10	0.15 12	0.22 3	0.45 3	0.18 12	0.19 30	0.88 25	0.12 32	0.60 3	0.90 5	**0.28** 1	0.75 40	1.19 16	0.50 40	0.21 62	0.24 71	0.40 63	0.46 4	1.02 13	0.62 5
SimpleFlow [51]	23.9	0.09 16	0.24 14	0.08 22	0.24 33	0.78 30	0.20 36	0.43 30	0.96 31	0.21 31	0.16 16	0.77 12	0.09 13	0.71 13	1.04 14	0.55 18	1.47 57	1.59 39	0.76 61	0.13 23	0.12 6	0.22 27	0.50 13	1.04 15	0.72 14
Occlusion-TV-L1 [67]	25.1	0.09 16	0.26 24	0.07 8	0.22 20	0.74 26	0.18 21	0.51 38	1.15 40	0.21 31	0.18 25	0.91 26	0.10 25	0.87 30	1.25 29	0.72 29	0.47 20	1.38 25	0.36 29	0.10 4	0.12 6	0.11 2	0.83 43	1.78 46	0.96 39
Adaptive [20]	27.0	0.09 16	0.26 24	**0.06** 1	0.23 30	0.78 30	0.18 21	0.54 40	1.19 45	0.21 31	0.18 25	0.91 26	0.10 25	0.88 33	1.25 29	0.73 32	0.50 25	1.28 20	0.31 12	0.14 30	0.16 48	0.22 27	0.65 28	1.37 27	0.79 24
Adapt-Window [34]	28.2	0.10 25	0.24 14	0.09 31	0.19 7	0.59 4	0.15 12	0.27 7	0.64 9	0.17 8	0.18 25	0.82 17	0.11 28	0.74 16	1.07 17	0.56 19	1.78 66	1.73 45	0.95 61	0.22 63	0.16 48	0.45 66	0.70 32	1.28 23	0.88 32
DPOF [18]	28.3	0.12 43	0.33 43	0.08 22	0.26 37	0.80 34	0.20 36	0.24 4	0.49 4	0.20 27	0.19 30	0.83 18	0.13 35	0.66 8	0.98 10	0.40 4	1.11 49	1.41 30	0.57 44	0.25 69	0.14 31	0.55 69	0.51 16	1.02 13	0.54 3
ACK-Prior [27]	29.0	0.11 32	0.25 18	0.09 31	0.18 3	0.59 4	0.13 4	0.27 7	0.64 9	0.16 6	0.15 8	0.78 13	0.09 13	0.82 24	1.14 21	0.71 28	1.90 68	1.90 53	0.99 64	0.23 67	0.17 54	0.49 68	0.77 39	1.44 29	0.91 34
Complementary OF [21]	29.5	0.11 32	0.28 31	0.10 41	0.18 3	0.63 10	0.12 3	0.31 20	0.75 20	0.18 12	0.19 30	0.97 33	0.12 32	0.97 44	1.31 40	1.00 49	1.78 66	1.73 45	0.87 59	0.11 9	0.12 6	0.22 27	0.68 29	1.48 30	0.95 37
ComplOF-FED-GPU [36]	31.0	0.11 32	0.29 35	0.10 41	0.21 17	0.78 30	0.14 8	0.32 22	0.79 24	0.17 8	0.19 30	0.99 34	0.11 28	0.89 34	1.29 35	0.73 32	1.25 51	1.74 48	0.64 48	0.14 30	0.13 19	0.30 53	0.64 26	1.50 32	0.83 27
Classic++ [32]	31.4	0.09 16	0.25 18	0.07 8	0.23 30	0.78 30	0.19 29	0.43 30	1.00 33	0.22 34	0.20 34	1.11 39	0.10 25	0.87 30	1.30 37	0.66 26	0.47 20	1.62 40	0.33 18	0.17 50	0.14 31	0.32 58	0.79 42	1.64 40	0.92 35
Aniso. Huber-L1 [22]	31.5	0.10 25	0.28 31	0.08 22	0.31 45	0.88 39	0.28 48	0.56 43	1.13 38	0.29 47	0.20 34	0.92 30	0.13 35	0.84 27	1.20 25	0.70 27	0.39 6	1.23 18	0.28 6	0.17 50	0.15 41	0.27 42	0.64 26	1.36 26	0.79 24
TriangleFlow [30]	34.2	0.11 32	0.29 35	0.09 31	0.26 37	0.95 43	0.17 18	0.47 37	1.07 36	0.18 12	0.16 16	0.87 24	0.09 13	1.07 52	1.47 57	1.10 53	0.87 43	1.39 27	0.57 44	0.15 38	0.19 64	0.23 31	0.63 25	1.33 24	0.84 28

Fig. 11 Screen shot of Middlebury EPE table at the writing of the paper (Sep. 2012). There are 73 methods in total and only the higher-ranking ones are shown.

Table 15 Average end-point error (EPE) on the Middlebury *training set* for the proposed new objective with the non-local term and alternating optimization (**Classic-C-A**) and its improved models. The statistical significance is tested using the Wilcoxon signed rank test between each method and the baseline (**Classic-C**).

	Average	Venus	Dimetrodon	Hydrangea	RubberWhale	Grove2	Grove3	Urban2	Urban3	signif.	p-value
Classic-C	0.298	0.281	0.152	0.165	0.093	0.158	0.627	0.348	0.562	—	—
Classic-C-A	0.305	0.281	0.140	0.159	0.092	0.167	0.676	0.334	0.594	0	0.8125
Classic-C-A-noRep	0.309	0.279	0.139	0.161	0.093	0.157	0.653	0.370	0.619	0	0.5781
Classic-C-A-II	0.296	0.278	0.153	0.166	0.091	0.168	0.656	0.329	0.531	0	0.7188
Classic-C-A-CGD	0.305	0.281	0.148	0.161	0.093	0.159	0.697	0.344	0.560	0	0.5625

31. B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conferences on Artificial Intelligence*, pages 674–679, 1981.
32. O. Mac Aodha, G. J. Brostow, and M. Pollefeys. Segmenting video into classes of algorithm-suitability. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1778–1785, 2010.
33. D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W.H. Freeman, New York, NY, USA, 1982.
34. H.-H. Nagel and W. Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Transaction on Pattern Analysis Machine Intelligence*, 8(5):565–593, September 1986.
35. M. Nikolova. Model distortions in Bayesian MAP reconstruction. *AIMS J. on Inverse Problems and Imaging*, 1:399–422, 2007.
36. W. H. Press, W. T. Vetterling, S. A. Teukolsky, and B. P. Flannery. *Numerical Recipes in C++: The art of scientific computing*. Cambridge University Press, New York, NY, USA, 2002.
37. X. Ren. Local grouping for optical flow. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.
38. L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, November 1992.
39. P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. *International Journal of Computer Vision*, 80(1):72–91, October 2008.
40. U. Schmidt, Q. Gao, and S. Roth. A generative perspective on MRFs in low-level vision. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1751–1758, 2010.
41. D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 2432–2439, 2010.
42. D. Sun, S. Roth, J. P. Lewis, and M. J. Black. Learning optical flow. In *European Conference on Computer Vision*, pages 83–97, 2008.
43. D. Sun, E. B. Sudderth, and M. J. Black. Layered image motion with explicit occlusions, temporal consistency, and depth ordering. In *NIPS*, pages 2226–2234, 2010.

Average angle error	avg. rank	Army (Hidden texture) GT im0 im1 all	Army disc	Army untext	Mequon (Hidden texture) GT im0 im1 all	Mequon disc	Mequon untext	Schefflera (Hidden texture) GT im0 im1 all	Schefflera disc	Schefflera untext	Wooden (Hidden texture) GT im0 im1 all	Wooden disc	Wooden untext	Grove (Synthetic) GT im0 im1 all	Grove disc	Grove untext	Urban (Synthetic) GT im0 im1 all	Urban disc	Urban untext	Yosemite (Synthetic) GT im0 im1 all	Yosemite disc	Yosemite untext	Teddy (Stereo) GT im0 im1 all	Teddy disc	Teddy untext
nLayers [60]	6.8	2.80 1	**7.42** 1	2.20 2	2.71 14	7.24 3	2.55 30	2.61 3	6.24 3	2.45 28	2.30 2	12.7 5	1.16 2	**2.30** 1	**3.02** 1	**1.70** 1	2.62 2	**6.95** 1	2.09 2	2.29 18	3.46 13	1.89 17	1.38 4	3.06 5	1.29 4
MDP-Flow2 [74]	7.0	3.23 15	7.93 3	2.60 8	**1.92** 1	**6.64** 1	**1.52** 1	2.46 2	5.91 2	1.56 2	3.05 18	15.8 19	1.51 17	2.77 8	3.50 4	2.16 8	2.86 4	8.58 3	2.70 10	2.00 9	3.50 18	1.59 9	1.28 2	2.67 2	0.89 2
ADF [71]	9.2	2.98 4	8.32 7	2.28 3	2.27 3	8.35 9	1.81 5	3.55 15	9.74 16	2.17 15	3.15 24	16.8 25	1.29 5	2.64 4	3.55 6	1.81 2	3.02 5	9.08 4	2.38 4	2.29 18	3.48 15	2.07 22	1.34 3	3.03 3	1.11 3
Layers++ [38]	11.2	3.11 6	8.22 5	2.79 19	2.43 7	7.02 2	2.24 16	**2.43** 1	**5.77** 1	2.18 17	**2.13** 1	**9.71** 1	**1.15** 1	2.35 2	**3.02** 1	1.96 4	3.81 22	11.4 18	3.22 26	2.74 35	4.01 38	2.35 28	1.45 8	3.05 4	1.79 10
IROF++ [61]	12.1	3.17 10	8.69 10	2.61 9	2.79 15	9.61 15	2.33 19	3.43 9	8.86 11	2.38 22	2.87 13	14.8 13	1.52 18	2.74 7	3.57 7	2.19 9	3.20 10	9.70 11	2.71 11	1.96 8	3.45 12	1.22 5	1.80 12	4.06 14	2.50 20
ALD-Flow [72]	12.2	2.82 2	7.86 2	**2.16** 1	2.84 18	10.1 19	1.86 8	3.73 17	10.4 19	1.67 4	3.10 20	16.8 25	1.28 4	2.69 6	3.60 8	1.85 3	2.79 3	11.3 17	2.32 3	2.07 10	3.25 5	3.10 49	2.03 18	5.11 20	1.94 12
Efficient-NL [64]	13.3	3.01 5	8.29 6	2.30 5	3.12 29	10.3 21	2.40 20	3.83 20	9.97 18	2.08 13	2.76 11	14.4 11	1.45 9	2.64 4	3.51 5	2.07 7	3.06 6	8.23 2	2.49 6	2.53 28	3.73 24	2.46 30	1.91 13	3.32 8	2.40 19
Sparse-NonSparse [58]	13.5	3.14 8	8.75 12	2.76 16	3.02 26	10.6 23	2.43 24	3.45 12	8.96 12	2.36 20	2.66 6	13.7 7	1.42 6	2.85 13	3.75 13	2.33 12	3.28 11	9.40 8	2.73 12	2.42 24	3.31 6	2.69 33	1.47 6	3.07 6	1.66 7
LSM [40]	14.6	3.12 7	8.62 9	2.75 15	3.00 25	10.5 22	2.44 25	3.43 9	8.85 10	2.35 19	2.66 6	13.6 6	1.44 7	2.82 10	3.68 9	2.36 14	3.38 14	9.41 9	2.81 16	2.69 33	3.52 19	2.84 37	1.59 8	3.38 10	1.80 11
TC-Flow [47]	14.7	2.91 3	8.00 4	2.34 6	2.18 2	8.77 10	**1.52** 1	3.84 21	10.7 23	**1.49** 1	3.13 21	16.6 24	1.46 12	2.78 9	3.73 12	1.96 4	3.08 7	11.4 18	2.66 7	1.94 6	3.43 11	3.20 53	3.06 26	7.04 26	4.08 46
Ramp [66]	15.2	3.18 12	8.83 14	2.73 14	2.89 21	10.1 19	2.44 25	3.27 7	8.43 7	2.38 22	2.74 10	14.2 9	1.46 12	2.82 10	3.69 11	2.29 10	3.37 13	9.31 7	2.93 18	2.62 31	3.38 10	3.19 52	1.54 7	3.21 7	2.24 16
COFM [62]	16.4	3.17 10	9.90 29	2.46 7	2.41 5	8.34 8	1.92 9	3.77 18	10.5 20	2.54 30	2.71 9	14.9 15	1.19 3	3.08 19	3.92 18	3.25 43	3.83 23	10.9 14	3.15 23	2.20 16	3.35 7	2.91 44	1.62 10	**2.56** 1	2.09 13
Classic+NL [31]	17.2	3.20 14	8.72 11	2.81 20	3.02 26	10.6 23	2.44 25	3.46 13	8.84 9	2.38 22	2.78 12	14.3 10	1.46 12	2.83 12	3.68 9	2.31 11	3.40 15	9.09 5	2.76 15	2.87 40	3.82 28	2.86 40	1.67 11	3.53 11	2.26 18
TV-L1-MCT [68]	17.4	3.16 9	8.48 8	2.71 13	3.28 33	10.8 29	2.60 33	3.95 23	10.5 20	2.38 22	2.69 8	13.9 8	1.45 9	2.94 15	3.79 14	2.63 27	3.50 18	9.75 12	3.06 21	2.08 11	3.35 7	2.29 26	1.95 15	3.89 13	2.71 23
SimpleFlow [51]	19.9	3.35 17	9.20 18	2.98 24	3.18 30	10.7 27	2.71 36	5.06 30	12.6 28	2.70 31	2.95 15	15.1 16	1.58 22	2.91 14	3.79 14	2.47 20	3.59 20	9.49 10	2.99 19	2.39 22	3.46 13	2.24 25	1.60 9	3.56 12	1.57 5
Direct ZNCC [70]	21.0	3.50 22	8.96 15	2.70 12	2.46 9	9.21 14	1.83 7	5.20 31	13.2 31	1.61 3	2.48 3	10.2 2	1.49 16	3.32 27	4.34 29	2.60 24	4.60 38	14.6 35	4.31 46	2.62 31	3.64 23	3.09 48	1.96 16	4.70 16	1.58 6
CostFilter [41]	21.3	3.84 27	9.64 26	3.06 26	2.55 12	8.09 6	2.03 12	2.69 4	6.47 4	1.88 8	3.66 33	16.8 25	1.88 31	2.62 3	3.34 3	1.99 6	4.05 28	11.0 16	3.65 35	4.16 61	7.18 72	4.66 62	**1.16** 1	3.36 9	**0.87** 1
OF-Mol [48]	21.8	3.19 13	8.76 13	2.77 17	3.84 40	14.0 44	2.69 35	3.44 11	8.78 8	2.39 26	2.98 16	15.8 19	1.53 19	2.96 16	3.89 17	2.34 13	3.40 15	9.30 6	2.73 12	2.83 38	3.92 33	2.98 47	2.46 20	4.98 19	2.89 26
MDP-Flow [26]	21.9	3.48 21	9.46 23	3.10 28	2.45 6	7.36 4	2.41 21	3.21 6	8.31 6	2.78 34	3.18 25	17.8 32	1.70 26	3.03 17	3.87 16	2.60 24	3.43 17	12.6 22	2.81 16	2.19 15	3.88 31	1.60 10	4.13 39	9.96 42	3.86 43
IROF-TV [55]	22.5	3.40 20	9.29 20	2.95 23	2.99 24	11.1 31	2.53 29	3.81 19	9.81 17	2.44 27	3.25 26	16.9 28	1.78 29	3.27 26	4.10 22	2.93 36	4.47 34	16.0 43	3.53 31	1.70 3	3.21 4	1.12 3	1.91 13	4.75 17	2.19 15
Sparse Occlusion [56]	24.7	3.62 24	9.12 16	2.90 21	2.92 22	9.08 13	2.56 31	4.49 26	11.8 26	2.11 14	3.14 22	15.8 19	1.57 21	3.26 25	4.22 27	2.36 14	3.52 19	10.9 14	2.66 7	5.10 70	6.32 68	3.15 51	2.02 17	4.92 18	1.71 8
OFH [39]	25.0	3.90 30	9.77 28	3.62 40	2.84 18	11.0 30	2.04 13	5.52 34	14.4 35	1.89 9	3.52 28	20.5 42	1.60 24	3.18 21	4.06 21	2.82 32	3.86 24	14.1 32	3.59 32	1.77 5	3.62 22	1.81 14	2.64 23	7.08 28	2.15 14
NL-TV-NCC [25]	25.0	3.89 29	9.16 17	2.98 24	2.87 20	9.69 16	1.99 10	4.44 25	11.6 25	1.76 6	2.64 4	11.8 3	1.48 15	3.49 39	4.60 44	2.47 20	4.67 42	13.5 27	4.26 45	2.83 38	4.57 51	2.84 37	2.62 22	6.00 24	2.25 17
TrajectoryFlow [59]	25.2	3.71 25	9.92 30	2.77 17	2.61 13	9.96 18	1.81 5	4.59 28	12.6 28	1.68 5	2.65 5	12.5 4	1.45 9	3.48 38	4.69 51	2.42 19	4.19 31	13.3 26	4.18 42	2.87 40	4.11 40	4.33 61	2.56 21	5.70 21	3.00 28
Occlusion-TV-L1 [67]	25.8	3.59 23	9.61 24	2.64 11	2.93 23	10.6 23	2.41 21	6.16 38	15.2 36	2.70 31	3.32 27	17.0 29	1.68 25	3.38 31	4.44 34	2.82 32	3.10 8	13.2 25	2.68 9	2.17 13	3.52 19	1.46 6	4.63 45	11.1 52	3.53 35
Complementary OF [21]	26.6	4.44 42	11.2 38	4.04 45	2.51 11	9.77 17	1.74 4	3.93 22	10.6 22	2.04 11	3.87 36	18.8 34	2.19 36	3.17 20	4.00 19	2.92 35	4.64 40	13.8 29	3.64 34	2.17 13	3.36 9	2.51 31	3.08 27	7.04 26	3.65 38
Adaptive [20]	27.6	3.29 16	9.43 22	2.28 3	3.10 28	11.4 33	2.46 28	6.58 40	15.7 41	2.52 29	3.14 22	15.6 17	1.56 20	3.67 46	4.46 36	3.48 48	3.32 12	13.0 24	2.38 4	2.76 37	4.39 48	1.93 19	3.58 33	8.18 32	2.88 25
ACK-Prior [27]	28.0	4.19 36	9.27 19	3.60 38	2.40 4	8.21 7	1.65 3	3.40 8	8.96 12	1.84 7	2.87 13	14.4 11	1.44 7	3.36 30	4.15 23	3.07 39	6.35 59	16.1 45	4.90 50	4.21 62	4.80 55	6.03 68	3.29 30	5.99 23	2.82 24
DPOF [18]	28.5	4.67 49	12.6 45	3.30 30	3.57 38	10.6 23	3.12 43	3.09 5	7.50 5	2.32 18	3.06 19	14.8 13	1.82 30	3.21 23	4.18 25	2.79 31	4.47 34	12.5 21	3.33 27	4.09 60	3.92 33	6.96 69	2.09 19	4.39 15	1.74 9
Adapt-Window [34]	30.4	4.07 33	9.32 21	3.54 37	2.42 6	7.97 5	1.99 10	3.47 14	8.99 14	2.05 12	3.55 30	17.0 29	1.97 32	3.34 28	4.21 26	2.82 32	5.93 52	14.8 37	4.83 49	4.32 63	4.61 53	5.39 66	3.27 29	5.89 22	3.16 29
ComplOF-FED-GPU [36]	31.8	4.28 39	11.3 39	3.70 41	3.25 31	13.0 38	2.16 14	4.06 24	11.2 24	1.95 10	3.91 37	19.2 36	2.01 33	3.20 22	4.15 23	2.64 28	4.61 39	16.1 45	3.90 37	2.98 45	3.77 26	3.69 57	2.85 24	7.44 30	2.53 21
Aniso. Huber-L1 [22]	33.1	3.71 25	10.1 31	3.08 27	4.36 46	13.0 38	3.77 47	6.92 43	15.3 38	3.60 45	3.54 29	15.9 22	2.04 34	3.38 31	4.45 35	2.47 20	3.88 25	12.9 23	2.74 14	3.37 50	4.36 47	2.85 39	3.16 28	7.52 31	2.90 27
Classic++ [32]	34.9	3.37 19	9.67 27	2.91 22	3.28 33	12.1 35	2.61 34	5.46 33	14.1 34	3.00 36	3.63 32	20.2 40	1.70 26	3.24 24	4.34 29	2.60 24	4.65 41	16.0 43	3.60 33	3.09 47	3.94 36	3.28 55	4.64 46	10.4 48	3.71 40
TV-L1-improved [17]	35.5	3.36 18	9.63 25	2.62 10	2.82 16	10.7 27	2.23 15	6.50 39	15.8 42	2.73 33	3.80 35	21.3 46	1.76 28	3.34 28	4.38 33	2.39 16	5.97 53	18.1 54	5.67 56	3.57 52	4.92 58	3.43 56	4.01 38	9.84 41	3.44 34

Fig. 12 Screen shot of Middlebury AAE table at the writing of the paper (Sep. 2012). There are 73 methods in total and only the higher-ranking ones are shown.

Table 16 Average end-point error (EPE) on the Middlebury *training set* for the proposed new objective with the weighted non-local term and its variants. The statistical significance is tested using the Wilcoxon signed rank test between each method and the baseline (**Classic+NL**).

	Average	Venus	Dimetrodon	Hydrangea	RubberWhale	Grove2	Grove3	Urban2	Urban3	signif.	*p*-value
Classic+NL	0.221	0.238	0.131	0.152	0.073	0.103	0.468	0.220	0.384	—	—
Classic+NL-Full	0.222	0.252	0.135	0.156	0.074	0.097	0.469	0.214	0.382	0	0.8203
Classic+NL-Fast	0.221	0.233	0.117	0.151	0.076	0.098	0.464	0.210	0.421	0	0.3125
RGB	0.240	0.243	0.131	0.155	0.081	0.109	0.501	0.236	0.468	**1**	**0.0156**
HSV	0.231	0.245	0.131	0.152	0.074	0.110	0.492	0.222	0.424	**1**	**0.0312**
LUV	0.226	0.241	0.131	0.149	0.074	0.104	0.460	0.223	0.427	0	0.5625
Gray	0.253	0.253	0.133	0.158	0.086	0.125	0.547	0.242	0.479	**1**	**0.0078**
w/o color	0.283	0.258	0.128	0.157	0.087	0.155	0.633	0.303	0.543	**1**	**0.0156**
w/o occ	0.226	0.243	0.131	0.152	0.073	0.103	0.488	0.230	0.386	0	0.1250
w/o spa	0.223	0.237	0.132	0.154	0.073	0.102	0.475	0.213	0.398	0	0.5625
$\sigma_2 = 5$	0.221	0.240	0.131	0.151	0.073	0.104	0.466	0.208	0.392	0	1.0000
$\sigma_2 = 10$	0.224	0.238	0.132	0.153	0.073	0.102	0.485	0.228	0.384	0	0.2500
$\lambda = 1$	0.236	0.245	0.151	0.164	0.080	0.120	0.430	0.243	0.459	0	0.1406
$\lambda = 9$	0.244	0.249	0.137	0.160	0.091	0.111	0.577	0.201	0.426	0	0.1016
11×11	0.223	0.240	0.131	0.151	0.074	0.103	0.451	0.234	0.397	0	0.5938
19×19	0.220	0.238	0.132	0.154	0.073	0.103	0.470	0.210	0.384	0	0.8750

44. R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.
45. T. Vaudrey and R. Klette. Residual images remove illumination artifacts! In *Pattern Recognition (Proceedings of DAGM)*, pages 472–481, Berlin, Heidelberg, 2009. Springer-Verlag.
46. J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, September 1994.
47. A. Wedel, T. Pock, J. Braun, U. Franke, and D. Cremers. Duality TV-L1 flow with fundamental matrix prior. In *Image and Vision Computing New Zealand*, 2008.
48. A. Wedel, T. Pock, and D. Cremers. Structure- and motion-adaptive regularization for high accuracy optic flow. In *IEEE International Conference on Computer Vision*, pages 1663–1668, 2009.
49. A. Wedel, T. Pock, C. Zach, D. Cremers, and H. Bischof. An improved algorithm for TV-L1 optical flow. In *Dagstuhl Motion Workshop*, pages 23–45, 2008.
50. M. Werlberger, T. Pock, and H. Bischof. Motion estimation with non-local total variation regularization. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 2464–2471, 2010.
51. M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 optical flow. In *BMVC*, pages

Table 17 Additional results for HS. Average end-point error (EPE) on the Middlebury *training set*. The statistical significance is tested using the Wilcoxon signed rank test between each method and **HS**.

	Average	Venus	Dimetrodon	Hydrangea	RubberWhale	Grove2	Grove3	Urban2	Urban3	signif.	*p*-value
HS	0.384	0.337	0.219	0.189	0.118	0.204	0.688	0.463	0.853	—	—
HS $20\times$ **MF**	0.365	0.299	0.214	0.184	0.104	0.196	0.699	0.431	0.792	**1**	**0.0469**

108.1–108.11, 2009.
52. F. Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83, December 1945.
53. J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi. Bilateral filtering-based optical flow estimation with occlusion detection. In *European Conference on Computer Vision*, volume I, pages 211–224, 2006.
54. L. Xu, J. Chen, and J. Jia. A segmentation based variational model for accurate optical flow estimation. In *European Conference on Computer Vision*, volume I, pages 671–684, 2008.
55. L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1293–1300, 2010.
56. K. Yoon and I. Kweon. Adaptive support-weight approach for correspondence search. *IEEE Transaction on Pattern Analysis Machine Intelligence*, 28(4):650–656, April 2006.
57. C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. In *Pattern Recognition (Proceedings of DAGM)*, pages 214–223, 2007.
58. H. Zimmer, A. Bruhn, and J. Weickert. Optic flow in harmony. *International Journal of Computer Vision*, 93(3):368–388, 2011.
59. H. Zimmer, A. Bruhn, J. Weickert, L. Valgaerts, A. Salgado, B. Rosenhahn, and H.-P. Seidel. Complementary optic flow. In *Energy Minimization Methods in Computer Vision and Pattern*, pages 207–220, 2009.
60. C. Zitnick, N. Jojic, and S. B. Kang. Consistent segmentation for optical flow estimation. In *IEEE International Conference on Computer Vision*, volume 2, pages 1308–1315, 2005.