A Database and Evaluation Methodology for Optical Flow

Simon Baker, Daniel Scharstein, J.P. Lewis,
Stefan Roth, Michael J. Black, and Richard Szeliski

December 2009
Technical Report
MSR-TR-2009-179

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://www.research.microsoft.com

Abstract

The quantitative evaluation of optical flow algorithms by Barron et al. (1994) led to significant advances in performance. The challenges for optical flow algorithms today go beyond the datasets and evaluation methods proposed in that paper. Instead, they center on problems associated with complex natural scenes, including nonrigid motion, real sensor noise, and motion discontinuities. We propose a new set of benchmarks and evaluation methods for the next generation of optical flow algorithms. To that end, we contribute four types of data to test different aspects of optical flow algorithms: (1) sequences with nonrigid motion where the ground-truth flow is determined by tracking hidden fluorescent texture, (2) realistic synthetic sequences, (3) high frame-rate video used to study interpolation error, and (4) modified stereo sequences of static scenes. In addition to the average angular error used by Barron et al., we compute the absolute flow endpoint error, measures for frame interpolation error, improved statistics, and results at motion discontinuities and in textureless regions. In October 2007, we published the performance of several well-known methods on a preliminary version of our data to establish the current state of the art. We also made the data freely available on the web at *http://vision.middlebury.edu/flow/*. Subsequently a number of researchers have uploaded their results to our website and published papers using the data. A significant improvement in performance has already been achieved. In this paper we analyze the results obtained to date and draw a large number of conclusions from them.

1 Introduction

As a subfield of computer vision matures, datasets for quantitatively evaluating algorithms are essential to ensure continued progress. Many areas of computer vision, such as stereo [63], face recognition [28, 55, 68], and object recognition [23, 24], have challenging datasets to track the progress made by leading algorithms and to stimulate new ideas. Optical flow was actually one of the first areas to have such a benchmark, introduced by Barron et al. in 1994 [7]. The field benefited greatly from this study which led to rapid and measurable progress. To continue the rapid progress, new and more challenging datasets are needed to push the limits of current technology, reveal where current algorithms fail, and evaluate the next generation of optical flow algorithms. Such an evaluation dataset for optical flow should ideally consist of complex real scenes with all the artifacts of real sensors (noise, motion blur, etc.). They should also contain substantial motion discontinuities and nonrigid motion. Of course, the image data should be paired with dense, subpixel-accurate, ground-truth flow fields.

The presence of nonrigid or independent motion makes collecting a ground-truth dataset for optical flow far harder than for stereo, say, where structured light [63] or range scanning [66] can be used to obtain ground truth. Our solution is to collect four different datasets, each satisfying a different subset of the desirable properties above. The combination of these datasets provides a basis for a thorough evaluation of current optical flow algorithms. Moreover, the relative performance of algorithms on the different datatypes may stimulate further research. In particular, we collected the following four types of data:

- **Real Imagery of Nonrigidly Moving Scenes:** Dense ground-truth flow is obtained using hidden fluorescent texture painted on the scene. We slowly move the scene, at each point capturing separate test images (in visible light) and ground-truth images (in UV light). Note that a related technique is being used commercially for motion capture [48] and Tappen et al. [73] recently used certain wavelengths to hide ground truth in intrinsic images. Another form of hidden markers was also used in [58] to provide a sparse ground-truth alignment (or flow) of face images. Finally, Liu et al. recently proposed a method to obtain ground-truth using human annotation [42].

- **Realistic Synthetic Imagery:** We address the limitations of simple synthetic sequences such as **Yosemite** [7] by rendering more complex scenes with larger motion ranges, more realistic texture, independent motion, and with more complex occlusions.

- **Imagery for Frame Interpolation:** Intermediate frames are withheld and used as ground truth. In a wide class of applications such as video re-timing, novel-view generation, and motion-compensated compression, what is important is not how well the flow matches the ground-truth motion, but how well intermediate frames can be predicted using the flow [72].

- **Real Stereo Imagery of Rigid Scenes:** Dense ground truth is captured using structured light [64]. The data is then adapted to be more appropriate for optical flow

by cropping to make the disparity range roughly symmetric.

We collected enough data to be able to split our collection into a training set (12 datasets) and a final evaluation set (12 datasets). The training set includes the ground truth and is meant to be used for debugging, parameter estimation, and possibly even learning [41, 70]. The ground truth for the final evaluation set is not publicly available (with the exception of the **Yosemite** sequence, which is included in the test set to allow some comparison with algorithms published prior to the release of our data).

We also extend the set of performance measures and the evaluation methodology of [7] to focus attention on current algorithmic problems:

- **Error Metrics:** We report both average angular error [7] and flow endpoint error (pixel distance) [53]. For image interpolation, we compute the residual RMS error between the interpolated image and the ground-truth image. We also report a gradient-normalized RMS error [72].

- **Statistics:** In addition to computing averages and standard deviations as in [7], we also compute robustness measures [63] and percentile-based accuracy measures [66].

- **Region Masks:** Following [63], we compute the error measures and their statistics over certain masked regions of research interest. In particular, we compute the statistics near motion discontinuities and in textureless regions.

In October 2007 we published the performance of several well-known algorithms on a preliminary version of our data to establish the current state of the art [6]. We also made the data freely available on the web at *http://vision.middlebury.edu/flow/*. Subsequently a large number of researchers have uploaded their results to our website and published papers using the data. A significant improvement in performance has already been achieved. In this paper we present both results obtained by the classic algorithms, as well as results obtained since publication of our preliminary data. In addition to summarizing the overall conclusions of the currently uploaded results, we also examine how the results vary: (1) across the metrics, statistics, and region masks, (2) across the various datatypes and datasets, (3) from flow estimation to interpolation, and (4) depending on the components of the algorithms.

The remainder of this paper is organized as follows. We begin in Section 2 with a survey of existing optical flow algorithms, benchmark databases, and evaluations. In Section 3 we describe the design and collection of our database, and briefly discuss the pros and cons of each dataset. In Section 4 we describe the evaluation metrics. In Section 5 we present the experimental results and discuss the major conclusions that can be drawn from them.

2 Related Work

Optical flow estimation is an extensive field. A fully comprehensive survey is beyond the scope of this paper. In this related work section, our goals are: (1) to present a taxonomy

of the main components in the majority of existing optical flow algorithms, and (2) to focus primarily on recent work and place the contributions of this work in the context of our taxonomy. Note that our taxonomy is similar to those of Stiller and Konrad [69] for optical flow and Scharstein and Szeliski [63] for stereo. For more extensive coverage of older work, the reader is referred to previous surveys such as those by Aggarwal and Nandhakumar [2], Barron et al. [7], Otte and Nagel [53], Mitiche and Bouthemy [47], and Stiller and Konrad [69].

We first define what we mean by optical flow. Following Horn's [32] taxonomy, the *motion field* is the 2D projection of the 3D motion of surfaces in the world, whereas the *optical flow* is the *apparent motion* of the brightness patterns in the image. These two motions are not always the same and, in practice, the goal of 2D motion estimation is application dependent. In frame interpolation, it is preferable to estimate apparent motion so that, for example, specular highlights move in a realistic way. On the other hand, in applications where the motion is used to interpret or reconstruct the 3D world, the motion field is what is desired.

In this paper, we consider both motion field estimation and apparent motion estimation, referring to them collectively as optical flow. The ground truth for most of our datasets is the true motion field, and hence this is how we define and evaluate optical flow accuracy. For our interpolation datasets, the ground truth consists of images captured at an intermediate time instant. For this data, our definition of optical flow is really the apparent motion.

We do, however, restrict attention to optical flow algorithms that estimate a separate 2D motion vector for each pixel in one frame of a sequence or video containing two or more frames. We exclude transparency which requires multiple motions per pixel. We also exclude more global representations of the motion such as parametric motion estimates [9].

Most existing optical flow algorithms pose the problem as the optimization of a global energy function that is the weighted sum of two terms:

$$E_{\text{Global}} = E_{\text{Data}} + \lambda E_{\text{Prior}}. \tag{1}$$

The first term E_{Data} is the *Data Term*, which measures how consistent the optical flow is with the input images. We consider the choice of the data term in Section 2.1. The second term E_{Prior} is the *Prior Term*, which favors certain flow fields over others (for example E_{Prior} often favors smoothly varying flow fields). We consider the choice of the prior term in Section 2.2. The optical flow is then computed by optimizing the global energy E_{Global}. We consider the choice of the optimization algorithm in Sections 2.3 and 2.4. In Section 2.5 we consider a number of miscellaneous issues. Finally, in Section 2.6 we survey previous databases and evaluations.

2.1 Data Term

2.1.1 Brightness Constancy

The basis of the data term used by most algorithms is *Brightness Constancy*, the assumption that when a pixel flows from one image to another, its intensity or color does not change. This assumption combines a number of assumptions about the reflectance properties of the

scene (e.g., that it is Lambertian), the illumination in the scene (e.g., that it is uniform [76]) and about the image formation process in the camera (e.g., that there is no vignetting). If $I(x, y, t)$ is the intensity of a pixel (x, y) at time t and the flow is $(u(x, y, t), v(x, y, t))$, Brightness Constancy can be written as:

$$I(x, y, t) \;=\; I(x + u, y + v, t + 1). \tag{2}$$

Linearizing Equation (2) by applying a Taylor expansion to the right hand side yields:

$$I(x, y, t) \;=\; I(x, y, t) + u\frac{\partial I}{\partial x} + v\frac{\partial I}{\partial y} + 1\frac{\partial I}{\partial t}, \tag{3}$$

which simplifies to the *Optical Flow Constraint* equation:

$$u\frac{\partial I}{\partial x} + v\frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} \;=\; 0. \tag{4}$$

Both Brightness Constancy and the Optical Flow Constraint equation provide just one constraint on the two unknowns at each pixel. This is the origin of the *Aperture Problem* and the reason that optical flow is ill-posed and must be regularized with a prior term (see Section 2.2).

The data term E_{Data} can be based on either Brightness Constancy in Equation (2) or on the Optical Flow Constraint in Equation (4). In either case, the equation is turned into an error per pixel, the set of which is then aggregated over the image in some manner (see Section 2.1.2). If Brightness Constancy is used, it is generally converted to the Optical Flow Constraint during the derivation of most continuous optimization algorithms (see Section 2.3), which often involves the use of a Taylor expansion to linearize the energies. The two constraints are therefore essentially equivalent in practical algorithms [54].

An alternative to the assumption of "constancy" is that the signals (images) at times t and $t+1$ are highly *correlated* [17, 57]. Various correlation constraints can be used for computing dense flow including normalized cross correlation and Laplacian correlation [18, 26].

2.1.2 Choice of the Penalty Function

Equations (2) and (4) both provide one error per pixel, which leads to the question of how these errors are aggregated over the image. A baseline approach is to use an L2 norm as in the Horn and Schunck algorithm [33]:

$$E_{\text{Data}} \;=\; \sum_{x,y} \left[u\frac{\partial I}{\partial x} + v\frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} \right]^2. \tag{5}$$

If Equation (5) is interpreted probabilistically, the use of the L2 norm means that the errors in the Optical Flow Constraint are assumed to be Gaussian and IID. This assumption is rarely true in practice, particularly near occlusion boundaries where pixels at time t may not be visible at time $t + 1$. In [11], Black and Anandan present an algorithm that can use

an arbitrary robust penalty function, illustrating their approach with the specific choice of a Lorentzian penalty function. A common choice by a number of recent algorithms [54, 80] is the L1 norm, which is sometimes approximated with the differentiable version:

$$\|x\|_1 \;\approx\; \sqrt{\|x\|_2^2 + \epsilon^2} \tag{6}$$

where $\|\cdot\|_1$ denotes the L1 norm, $\|\cdot\|_2$ denotes the L2 norm, and ϵ is a small positive constant. A variety of other penalty functions have been used.

2.1.3 Photometrically Invariant Features

Instead of using the raw intensity or color values in the images, it is also possible to use features computed from those images. In fact, some of the earliest optical flow algorithms used filtered images to reduce the effects of shadows [3, 18]. One recently popular choice (for example used in [54] among others) is to augment or replace Equation (2) with a similar term based on the gradient of the image:

$$\nabla I(x, y, t) \;=\; \nabla I(x + u, y + v, t + 1). \tag{7}$$

Empirically the gradient is often more robust to (approximately additive) illumination changes than the raw intensities. Note, however, that Equation (7) makes the additional assumption that the flow is locally translational; e.g., local scale changes, rotations, etc, can violate Equation (7) even when Equation (2) holds. Zimmer et al. [84] combine Brightness Constancy with Gradient Constancy and perform a careful normalization of the resulting data term. It is also possible to use more complicated features than the gradient. For example a Field-of-Experts formulation was used in [70] and SIFT features were used in [43].

2.1.4 Modeling Illumination, Blur, and Other Appearance Changes

The motivation for using features is to increase robustness to illumination and other appearance changes. Another approach is to estimate the change explicitly. For example, suppose $g(x, y)$ denotes a multiplicative scale factor and $b(x, y)$ an additive term that together model the illumination change between $I(x, y, t)$ and $I(x, y, t + 1)$. Brightness Constancy in Equation (2) can be generalized to:

$$g(x, y)I(x, y, t) \;=\; I(x + u, y + v, t + 1) + b(x, y). \tag{8}$$

Note that putting $g(x, y)$ on the left hand side is preferable to putting it on the right hand side as it can make optimization easier [65]. Equation (8) is even more under-constrained than Equation (2), with four unknowns per pixel rather than two. It can, however, be solved by putting an appropriate prior on the two components of the illumination change model $g(x, y)$ and $b(x, y)$ [51, 65]. Explicit illumination modeling can be generalized in several ways, for example to model the changes physically over a longer time interval [30] or to model blur [65].

2.2 Prior Term

The data term alone is ill-posed with fewer constraints than unknowns. It is therefore necessary to add a prior to favor one possible solution over another. Generally speaking, while most priors are smoothness priors, a wide variety of choices are possible.

2.2.1 First Order

Arguably the simplest prior is to favor small first-order derivatives (gradients) of the flow field. If we use an L2 norm, then we might, for example, define:

$$E_{\text{Prior}} = \sum_{x,y} \left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \right]. \tag{9}$$

The combination of Equations (5) and (9) defines the energy used by Horn and Schunck [33]. Given more than two frames in the video, it is also possible to add temporal smoothness terms $\frac{\partial u}{\partial t}$ and $\frac{\partial v}{\partial t}$ to Equation (9) [10, 49, 54]. Note, however, that the temporal terms need to be weighted differently from the spatial ones.

2.2.2 Choice of the Penalty Function

As for the data term in Section 2.1.2, under a probabilistic interpretation, the use of an L2 norm assumes that the gradients of the flow field are Gaussian and IID. Again, this assumption is violated in practice and so a wide variety of other penalty functions have been used. The Black and Anandan algorithm [11] also uses a first-order prior, but can use an arbitrary robust penalty function on the prior term rather than the L2 norm in Equation (9). While Black and Anandan [11] use the same Lorentzian penalty function for both the data and spatial term, there is no need for them to be the same. The L1 norm is also a popular choice of penalty function [54, 80]. When the L1 norm is used to penalize the gradients of the flow field, the formulation falls in the class of Total Variation (TV) methods.

There are two common ways such robust penalty functions are used. One approach is to apply the penalty function separately to each derivative and then to sum up the results. The other approach is to first sum up the squares (or absolute values) of the gradients and then apply a single robust penalty function. Some algorithms use the first approach [11], while others use the second [16, 54, 80].

Note that some penalty (log probability) functions have probabilistic interpretations related to the distribution of flow derivatives [61].

2.2.3 Spatial Weighting

One popular refinement for the prior term is one that weights the penalty function with a spatially varying function. One particular example is to vary the weight depending on the

gradient of the image:

$$E_{\text{Prior}} = \sum_{x,y} w(\nabla I) \left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \right]. \tag{10}$$

Equation (10) could be used to reduce the weight of the prior at edges (high $|\nabla I|$) because there is a greater likelihood of a flow discontinuity at an intensity edge than inside a smooth region. The weight can also be a function of an over-segmentation of the image, rather than the gradient, for example down-weighting the prior between different segments [65].

2.2.4 Anisotropic Smoothness

In Equation (10) the weighting function is isotropic, treating all directions equally. A variety of approaches weight the smoothness prior anisotropically. For example, Nagel and Enkelmann [50] and Werlberger et al. [82] weight the direction along the image gradient less than the direction orthogonal to it, and Sun et al. [70] learn a Steerable Random field to define the weighting. Zimmer et al. [84] perform a similar anisotropic weighting, but the directions are defined by the data constraint rather than the image gradient.

2.2.5 Higher-Order Priors

The first-order priors in Section 2.2.1 can be replaced with priors that encourage the second-order derivatives ($\frac{\partial^2 u}{\partial x^2}$, $\frac{\partial^2 u}{\partial y^2}$, $\frac{\partial^2 u}{\partial x \partial y}$, $\frac{\partial^2 v}{\partial x^2}$, $\frac{\partial^2 v}{\partial y^2}$, $\frac{\partial^2 v}{\partial x \partial y}$) to be small [4, 75].

A related approach is to use an affine prior [35, 36, 52, 65]. One approach is to over-parameterize the flow [52]. Instead of solving for two flow vectors $(u(x,y,t), v(x,y,t))$ at each pixel, the algorithm in [52] solves for 6 affine parameters $a_i(x,y,t), i = 1, \ldots 6$ where the flow is given by:

$$u(x,y,t) = a_1(x,y,t) + \frac{x - x_0}{x_0} a_3(x,y,t) + \frac{y - y_0}{y_0} a_5(x,y,t) \tag{11}$$

$$v(x,y,t) = a_2(x,y,t) + \frac{x - x_0}{x_0} a_4(x,y,t) + \frac{y - y_0}{y_0} a_6(x,y,t) \tag{12}$$

where (x_0, y_0) is the middle of the image. Equations (11) and (12) are then substituted into any of the data terms above. Ju et al. formulate the prior so that neighboring affine parameters should be similar [36]. As above, a robust penalty may be used and, further, may vary depending on the affine parameter (for example weighting a_1 and a_2 differently from $a_3 \ldots a_6$).

2.2.6 Rigidity Priors

A number of authors have explored rigidity or fundamental matrix priors which, in the absence of other evidence, favor flows that are parallel to epipolar lines. These constraints have both been strictly enforced [1, 29, 52] and added as a soft prior [78, 79].

2.3 Continuous Optimization Algorithms

The two most commonly used continuous optimization techniques in optical flow are: (1) gradient descent algorithms (Section 2.3.1) and (2) extremal or variational approaches (Section 2.3.2). In Section 2.3.3 we describe a small number of other approaches.

2.3.1 Gradient Descent Algorithms

Let $\mathbf{f}$ be a vector resulting from concatenating the horizontal and vertical components of the flow at every pixel. The goal is then to optimize E_{Global} with respect to $\mathbf{f}$. The simplest gradient descent algorithm is *steepest descent* [5], which takes steps in the direction of the negative gradient $-\frac{\partial E_{\text{Global}}}{\partial \mathbf{f}}$. An important question with steepest descent is how big the step size should be. One approach is to adjust the step size iteratively, increasing it if the algorithm makes a step that reduces the energy and decreasing it if the algorithm tries to makes a step that increases the error. Another approach used in [11] is to set the step size to be:

$$-w\frac{1}{T}\frac{\partial E_{\text{Global}}}{\partial \mathbf{f}}. \tag{13}$$

In this expression, T is an upper bound on the second derivatives of the energy; $T \geq \frac{\partial^2 E_{\text{Global}}}{\partial f_i^2}$ for all components f_i in the vector $\mathbf{f}$. The parameter $0 < w < 2$ is an over-relaxation parameter. Without it, Equation (13) tends to take too small steps because: (1) T is an upper bound, and (2) the equation does not model the off-diagonal elements in the Hessian. It can be shown that if E_{Global} is a quadratic energy function (i.e., the problem is equivalent to solving a large linear system), convergence to the global minimum can be guaranteed (albeit possibly slowly) for any $0 < w < 2$. In general E_{Global} is nonlinear and so there is no such guarantee. However, based on the theoretical result in the linear case, a value around $w \approx 1.95$ is generally used. Also note that many non-quadratic (e.g., robust) formulations can be solved with iteratively reweighted least squares (IRLS); i.e., they are posed as a sequence of quadratic optimization problems with a data-dependent weighting function that varies from iteration to iteration. The weighted quadratic is iteratively solved and the weights re-estimated.

In general, steepest descent algorithms are relatively weak optimizers requiring a large number of iterations because they fail to model the coupling between the unknowns. A second-order model of this coupling is contained in the Hessian matrix $\frac{\partial^2 E_{\text{Global}}}{\partial f_i \partial f_j}$. Algorithms that use the Hessian matrix or approximations to it such as the Newton method, Quasi-Newton methods, the Gauss-Newton method, and the Levenberg-Marquardt algorithm [5] all converge far faster. These algorithms are however inapplicable to the general optical flow problem because they require estimating and inverting the Hessian, a $2n \times 2n$ matrix where there are n pixels in the image. These algorithms are applicable to problems with fewer parameters such as the Lucas-Kanade algorithm [44] and variants [38], which solve for a single flow vector (2 unknowns) independently for each block of pixels. Another set of examples are parametric motion algorithms [9], which also just solve for a small number of unknowns.

2.3.2 Variational and Other Extremal Approaches

The second class of algorithms assume that the global energy function can be written in the form:

$$E_{\text{Global}} = \int\int E(u(x,y), v(x,y), x, y, u_x, u_y, v_x, v_y)\,\mathrm{d}x\,\mathrm{d}y \tag{14}$$

where $u_x = \frac{\partial u}{\partial x}$, $u_y = \frac{\partial u}{\partial y}$, $v_x = \frac{\partial v}{\partial x}$, and $v_y = \frac{\partial v}{\partial y}$. At this stage, $u = u(x,y)$ and $v = v(x,y)$ are treated as unknown 2D functions rather than the set of unknown parameters (the flows at each pixel). The parameterization of these functions occurs later. Note that Equation (14) imposes limitations on the functional form of the energy, i.e., that it is just a function of the flow u, v, the spatial coordinates x, y and the gradients of the flow u_x, u_y, v_x and v_y. A wide variety of energy functions do satisfy this requirement including [16, 33, 52, 54, 84].

Equation (14) is then treated as a "calculus of variations" problem leading to the Euler-Lagrange equations:

$$\frac{\partial E_{\text{Global}}}{\partial u} - \frac{\partial}{\partial x}\frac{\partial E_{\text{Global}}}{\partial u_x} - \frac{\partial}{\partial y}\frac{\partial E_{\text{Global}}}{\partial u_y} = 0 \tag{15}$$

$$\frac{\partial E_{\text{Global}}}{\partial v} - \frac{\partial}{\partial x}\frac{\partial E_{\text{Global}}}{\partial v_x} - \frac{\partial}{\partial y}\frac{\partial E_{\text{Global}}}{\partial v_y} = 0. \tag{16}$$

Because they use the calculus of variations, such algorithms are generally referred to as *variational*. In the special case of the Horn-Schunck algorithm [32], the Euler-Lagrange equations are linear in the unknown functions u and v. These equations are then parameterized with two unknown parameters per pixel and can be solved as a sparse linear system. A variety of options are possible, including the Jacobi method, the Gauss-Seidel method, Successive Over-Relaxation, and the Conjugate Gradient algorithm.

For more general energy functions, the Euler-Lagrange equations are nonlinear and are typically solved using an iterative method (analogous to gradient descent). For example, the flows can be parameterized by $u + \mathrm{d}u$ and $v + \mathrm{d}v$ where u, v are treated as known (from the previous iteration or the initialization) and $\mathrm{d}u$, $\mathrm{d}v$ as unknowns. These expressions are substituted into the Euler-Lagrange equations, which are then linearized through the use of Taylor expansions. The resulting equations are linear in $\mathrm{d}u$ and $\mathrm{d}v$ and solved using a sparse linear solver. The estimates of u and v are then updated appropriately and the next iteration applied.

One disadvantage of variational algorithms is that the discretization of the Euler-Lagrange equations is not always exact with respect to the original energy [56]. Another extremal approach [70], closely related to the variation algorithms is to use:

$$\frac{\partial E_{\text{Global}}}{\partial \mathbf{f}} = 0 \tag{17}$$

rather than the Euler-Lagrange equations. Otherwise, the approach is similar. Equation (17) can be linearized and solved using as a sparse linear system. The key difference between this approach and the variational one is just whether the parameterization of the flow functions

into a set of flows per pixel occurs before or after the derivation of the extremal constraint equation (Equation (17) or the Euler-Lagrange equations). One advantage of the early parameterization and the subsequent use of Equation (17) is that it reduces the restrictions on the functional form of E_{Global}, important in learning-based approaches [70].

2.3.3 Other Continuous Algorithms

Another approach [74, 80] is to decouple the data and prior terms through the introduction of two sets of flow parameters, say $(u_{\text{data}}, v_{\text{data}})$ for the data term and $(u_{\text{prior}}, v_{\text{prior}})$ for the prior:

$$\begin{aligned} E_{\text{Global}} &= E_{\text{Data}}(u_{\text{data}}, v_{\text{data}}) + \lambda E_{\text{Prior}}(u_{\text{prior}}, v_{\text{prior}}) \\ &\quad + \gamma \left(\|u_{\text{data}} - u_{\text{prior}}\|^2 + \|v_{\text{data}} - v_{\text{prior}}\|^2 \right). \end{aligned} \tag{18}$$

The final term in Equation (18) encourages the two sets of flow parameters to be roughly the same. For a sufficiently large value of γ the theoretical optimal solution will be unchanged and $(u_{\text{data}}, v_{\text{data}})$ will exactly equal $(u_{\text{prior}}, v_{\text{prior}})$. Practical optimization with too large a value of γ is problematic, however. In practice either a lower value is used or γ is steadily increased. The two sets of parameters allow the optimization to be broken into two steps. In the first step, the sum of the data term and the third term in Equation (18) is optimized over the data flows $(u_{\text{data}}, v_{\text{data}})$ assuming the prior flows $(u_{\text{prior}}, v_{\text{prior}})$ are constant. In the second step, the sum of the prior term and the third term in Equation (18) is optimized over prior flows $(u_{\text{prior}}, v_{\text{prior}})$ assuming the data flows $(u_{\text{data}}, v_{\text{data}})$ are constant. The result is two much simpler optimizations. The first optimization can be performed independently at each pixel. The second optimization is often simpler because it does not depend directly on the nonlinear data term [74, 80].

Finally, in recent work, continuous convex optimization algorithms such as Linear Programming have also been used to compute optical flow [65].

2.3.4 Coarse-To-Fine and Other Heuristics

All of the above algorithms solve the problem as huge nonlinear optimizations. Even the Horn-Schunck algorithm, which results in linear Euler-Lagrange equations, is nonlinear through the linearization of the Brightness Constancy constraint to give the Optical Flow constraint. A variety of approaches have been used to improve the convergence rate and reduce the likelihood of falling into a local minimum.

One component in many algorithms is a coarse-to-fine strategy. The most common approach is to build image pyramids by repeated blurring and downsampling [3, 11, 18, 22, 26, 44]. Optical flow is first computed on the top level (fewest pixels) and then upsampled and used to initialize the estimate at the next level. Computation at the higher levels in the pyramid involves far fewer unknowns and so is far faster. The initialization at each level from the previous level also means that far fewer iterations are required at each level. For this reason, pyramid algorithms tend to be significantly faster than a single solution

at the bottom level. The images at the higher levels also contain fewer higher frequency components reducing the number of local minima in data term. A related approach is to use a multi-grid algorithm [16] where estimates of the flow are passed both up and down the hierarchy of approximations. The major limitation of many coarse-to-fine algorithms, however, is the tendency to over-smooth fine structure and fast-moving objects.

The main purpose of coarse-to-fine strategies is to deal with nonlinearities caused by the data term (and the subsequent difficulty in dealing with long-range motion). At the coarsest pyramid level, the flow magnitude is likely to be small making the linearization of the brightness constancy assumption reasonable. Incremental warping of the flow between pyramid levels [9] helps keep the flow update at any given level small (i.e., under one pixel). When combined with incremental warping and updating within a level, this method is effective for optimization with a linearized brightness constancy assumption.

Another common cause of nonlinearity is the use of a robust penalty function (see Sections 2.1.2 and 2.2.2). A common approach to improve robustness in this case is Graduated Non-Convexity (GNC) [11, 13]. During GNC, the problem is first converted into a convex approximation that is more easily solved. The energy function is then made incrementally more non-convex and the solution is refined, until the original desired energy function is reached.

2.4 Discrete Optimization Algorithms

A number of recent approaches use discrete optimization algorithms, similar to those employed in stereo matching, such as graph cuts [14] and belief propagation [71]. Discrete optimization methods approximate the continuous space of solutions with a greatly simplified problem. The hope is that this will enable a more thorough and complete search of the state space. The trade-off in moving from continuous to discrete optimization is one of search efficiency for fidelity. Note that, in contrast to discrete stereo optimization methods, the 2D flow field makes discrete optimization of optical flow significantly more challenging. Approximations are usually made, which can limit the power of the discrete algorithms to avoid local minima. The few methods proposed to date can be divided into two main approaches described below.

2.4.1 Fusion Approaches

Algorithms such as [37, 40, 74] assume that a number of candidate flow fields have been generated by running standard algorithms such as Lucas-Kanade [44] and Horn-Schunck [33], possibly multiple times with a number of different parameters. Computing the flow is then posed as choosing which of the set of possible candidates is best at each pixel. Fusion Flow [40] uses a sequence of binary graph-cut optimizations to refine the current flow estimate by selectively replacing portions with one of the candidate solutions. Trobin et al. [74] perform a similar sequence of fusion steps, at each step solving a continuous $[0, 1]$ optimization problem and then thresholding the results.

2.4.2 Dynamically Reparameterizing Sparse State-Spaces

Any fixed 2D discretization of the continuous space of 2D flow fields is likely to be a crude approximation to the continuous field. A number of algorithms take the approach of first approximating this state space sparsely (both spatially, and in terms of the possible flows at each pixel) and then refining the state space based on the result. An early use of this idea for flow estimation employed simulated annealing with a state space that adapted based on the local shape of the objective function [10]. More recently, Glocker et al. [27] initially use a sparse sampling of possible motions on a coarse version of the problem. As the algorithm runs from coarse to fine, the spatial density of motion states (which are interpolated with a spline) and the density of possible flows at any given control point are chosen based on the uncertainty in the solution from the previous iteration. The algorithm of Lei et al. [39] also sparsely allocates states across space and for the possible flows at each spatial location. The spatial allocation uses a hierarchy of segmentations, with a single possible flow for each segment at each level. Within any level of the segmentation hierarchy, first a sparse sampling of the possible flows is used, followed by a denser sampling with a reduced range around the solution from the previous iteration. The algorithm in [20] iteratively alternates between two steps. In the first step, all the states are allocated to the horizontal motion, which is estimated similarly to stereo, assuming the vertical motion is zero. In the second step, all the states are allocated to the vertical motion, treating the estimate of the horizontal motion from the previous iteration as constant.

2.4.3 Continuous Refinement

An optional step after a discrete algorithm is to use a continuous optimization to refine the results. Any of the approaches in Section 2.3 are possible.

2.5 Miscellaneous Issues

2.5.1 Learning

The design of a global energy function E_{Global} involves a variety of choices, each with a number of free parameters. Rather than manually making these decision and tuning parameters, learning algorithms have been used to choose the data and prior terms and optimize their parameters by maximizing performance on a set of training data [41, 61, 70].

2.5.2 Segmentation

If the image can be segmented into coherently moving regions, many of the methods above can be used to accurately estimate the flow within the regions. Further, if the flow were accurately known, segmenting it into coherent regions would be feasible. One of the reasons optical flow has proven challenging to compute is that the flow and its segmentation must be computed together.

Several methods first segment the scene using non-motion cues and then estimate the flow in these regions [12,83]. Within each image segment, Black and Jepson [12] use a parametric model (e.g., affine) [9], which simplifies the problem by reducing the number of parameters to be estimated. The flow is then refined as suggested above.

2.5.3 Layers

Motion transparency has been extensively studied and is not considered in detail here. Most methods have focused on the use of parametric models that estimate motion in layers [34,77]. The regularization of transparent motion in the framework of global energy minimization, however, has received little attention with the exception of [36,81].

2.5.4 Sparse-to-Dense Approaches

The coarse-to-fine methods described above have difficulty dealing with long-range motion of small objects. In contrast, there exist many methods to accurately estimate sparse feature correspondences even when the motion is large. Such sparse matching method can be combined with the continuous energy minimization approaches in a variety of ways [15,43,60,83].

2.5.5 Visibility and Occlusion

Occlusions and visibility changes can cause major problems for optical flow algorithms. The most common solution is to model such effects implicitly using a robust penalty function on both the data term and the prior term. Explicit occlusion estimation, for example through cross-checking flows computed forwards and backwards in time, is another approach that can be used to improve robustness to occlusions and visibility changes [39,83].

2.6 Databases and Evaluations

Prior to our evaluation [6], there were three major attempts to quantitatively evaluate optical flow algorithms, each proposing sequences with ground truth. The work of Barron et al. [7] has been so influential that until recently, essentially all published methods compared with it. The synthetic sequences used there, however, are too simple to make meaningful comparisons between modern algorithms. Otte and Nagel [53] introduced ground truth for a real scene consisting of polyhedral objects. While this provided real imagery, the images were extremely simple. More recently, McCane et al. [46] provided ground truth for real polyhedral scenes as well as simple synthetic scenes. Most recently Liu et al. [42] proposed a dataset of real imagery that uses hand segmentation and computed flow estimates within the segmented regions to generate the ground truth. While this has the advantage of using real imagery, the reliance on human judgement for segmentation, and on a particular optical flow algorithm for ground truth, may limit its applicability.

In this paper we go beyond these studies in several important ways. First, we provide ground-truth motion for much more complex real and synthetic scenes. Specifically, we

include ground truth for scenes with nonrigid motion. Second, we also provide ground-truth motion boundaries and extend the evaluation methods to these areas where many flow algorithms fail. Finally, we provide a web-based interface, which facilitates the ongoing comparison of methods.

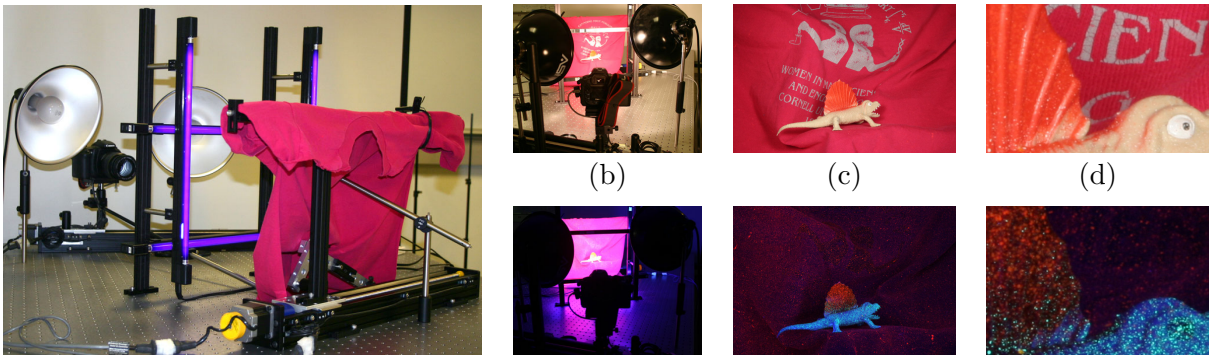
Our goal is to push the limits of current methods and, by exposing where and how they fail, focus attention on the hard problems. As described above, almost all flow algorithms have a specific data term, prior term, and optimization algorithm to compute the flow field. Regardless of the choices made, algorithms must somehow deal with all of the phenomena that make optical flow intrinsically ambiguous and difficult. These include: (1) the aperture problem and textureless regions, which highlight the fact that optical flow is inherently ill-posed, (2) camera noise, nonrigid motion, motion discontinuities, and occlusions, which make choosing appropriate penalty functions for both the data and prior terms important, (3) large motions and small objects which, often cause practical optimization algorithms to fall into local minima, and (4) mixed pixels, changes in illumination, non-Lambertian reflectance, and motion blur, which highlight overly simplified assumptions made by Brightness Constancy (or simple filter constancy). Our goal is to provide ground-truth data containing all of these components and to provide information about the location of motion boundaries and textureless regions. In this way, we hope to be able to evaluate which phenomena pose problems for which algorithms.

3 Database Design

Creating a ground-truth database for optical flow is difficult. For stereo, structured light [63] or range scanning [66] can be used to obtain dense, pixel-accurate ground truth. For optical flow, the scene may be moving nonrigidly making such techniques inapplicable in general. Ideally we would like imagery collected in real-world scenarios with real cameras and substantial nonrigid motion. We would also like dense, subpixel-accurate ground truth. We are not aware of any technique that can simultaneously satisfy all of these goals.

Rather than collecting a single type of data (with its inherent limitations) we instead collected four different types of data, each satisfying a different subset of desirable properties. Having several different types of data has the benefit that the overall evaluation is less likely to be affected by any biases or inaccuracies in any of the data types. It is important to keep in mind that no ground-truth data is perfect. The term itself just means "measured on the ground" and any measurement process may introduce noise or bias. We believe that the combination of our four datasets is sufficient to allow a thorough evaluation of current optical flow algorithms. Moreover, the relative performance of algorithms on the different types of data is itself interesting and can provide insights for future algorithms (see Section 5.2.4).

Wherever possible, we collected eight frames with the ground-truth flow being defined between the middle pair. We collected color imagery, but also make grayscale imagery available for comparison with legacy implementations and existing approaches that only process grayscale. The dataset is divided into 12 training sequences with ground truth, which can be used for parameter estimation or learning, and 12 test sequences, where the



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

Figure 1: (a) The setup for obtaining ground-truth flow using hidden fluorescent texture includes computer-controlled lighting to switch between the UV and visible lights. It also contains motion stages for both the camera and the scene. (b–d) The setup under the visible illumination. (e–g) The setup under the UV illumination. (c) and (f) show the high-resolution images taken by the digital camera. (d) and (g) show a zoomed portion of (c) and (f). The high-frequency fluorescent texture in the images taken under UV light (g) allows accurate tracking, but is largely invisible in the low-resolution test images.

ground truth is withheld. In this paper we only describe the test sequences. The datasets, instructions for evaluating results on the test set, and the performance of current algorithms are all available at *http://vision.middlebury.edu/flow/*. We describe each of the four types of data below.

3.1 Dense GT Using Hidden Fluorescent Texture

We have developed a technique for capturing imagery of nonrigid scenes with ground-truth optical flow. We build a scene that can be moved in very small steps by a computer-controlled motion stage. We apply a fine spatter pattern of fluorescent paint to all surfaces in the scene. The computer repeatedly takes a pair of high-resolution images both under ambient lighting and under UV lighting, and then moves the scene (and possibly the camera) by a small amount.

In our current setup, shown in Figure 1(a), we use a Canon EOS 20D camera to take images of size 3504×2336, and make sure that no scene point moves by more than 2 pixels from one captured frame to the next. We obtain our test sequence by downsampling every 40th image taken under visible light by a factor of six, yielding images of size 584×388. Because we sample every 40th frame, the motion can be quite large (up to 12 pixels between frames in our evaluation data) even though the motion between each pair of captured frames is small and the frames are subsequently downsampled, i.e., after the downsampling, the motion between any pair of captured frames is at most 1/3 of a pixel.

Since fluorescent paint is available in a variety of colors, the color of the objects in the scene can be closely matched. In addition, it is possible to apply a fine spatter pattern,

where individual droplets are about the size of 1–2 pixels in the high-resolution images. This high-frequency texture is therefore far less perceptible in the low-resolution images, while the fluorescent paint is very visible in the high-resolution UV images in Figure 1(g). Note that fluorescent paint absorbs UV light but emits light in the visible spectrum. Thus, the camera optics affect the hidden texture and the scene colors in exactly the same way, and the hidden texture remains perfectly aligned with the scene.

The ground-truth flow is computed by tracking small windows in the original sequence of high-resolution UV images. We use a sum-of-squared-difference (SSD) tracker with a window size of 15×15, corresponding to a window radius of less than 1.5 pixels in the downsampled images. We perform a local brute-force search, using each frame to initialize the next. We also crosscheck the results by tracking each pixel both forwards and backwards through the sequence and require perfect correspondence. The chances that this check would yield false positives after tracking for 40 frames are very low. Crosschecking identifies the occluded regions, whose motion we mark as "unknown." After the initial integer-based motion tracking and crosschecking, we estimate the subpixel motion of each window using Lucas-Kanade [44] with a precision of about 1/10 pixels (i.e., 1/60 pixels in the downsampled images). Using the combination of fluorescent paint, downsampling high-resolution images, and sequential tracking of small motions, we are able to obtain dense, subpixel accurate ground truth for a nonrigid scene.

We include four sequences in the evaluation set (Figure 2). **Army** contains several independently moving objects. **Mequon** contains nonrigid motion and large areas with little texture. **Schefflera** contains thin structures, shadows, and foreground/background transitions with little contrast. **Wooden** contains rigidly moving objects with little texture in the presence of shadows. The maximum motion in **Army** is approximately 4 pixels. The maximum motion in the other three sequences is about 10 pixels. All sequences are significantly more difficult than the **Yosemite** sequence due to the larger motion ranges, the non-rigid motion, various photometric effects such as shadows and specularities, and the detailed geometric structure.

The main benefit of this dataset is that it contains ground truth on imagery captured with a real camera. Hence, it contains real photometric effects, natural textural properties, etc. The main limitations of this dataset are that the scenes are laboratory scenes, not real-world scenes. There is also no motion blur due to the stop motion method of capture.

One drawback of this data is that the ground truth it is not available in areas where cross-checking failed, in particular, in regions occluded in one image. Even though the ground truth is reasonably accurate (on the order of 1/60th of a pixel), the process is not perfect; significant errors however, are limited to a small fraction of the pixels. The same can be said for any real data where the ground truth is measured, including, for example, in the Middlebury stereo dataset [63]. The ground-truth measuring technique may always be prone to errors and biases. Consequently, the following section describes realistic synthetic data where the ground truth is guaranteed to be perfect.

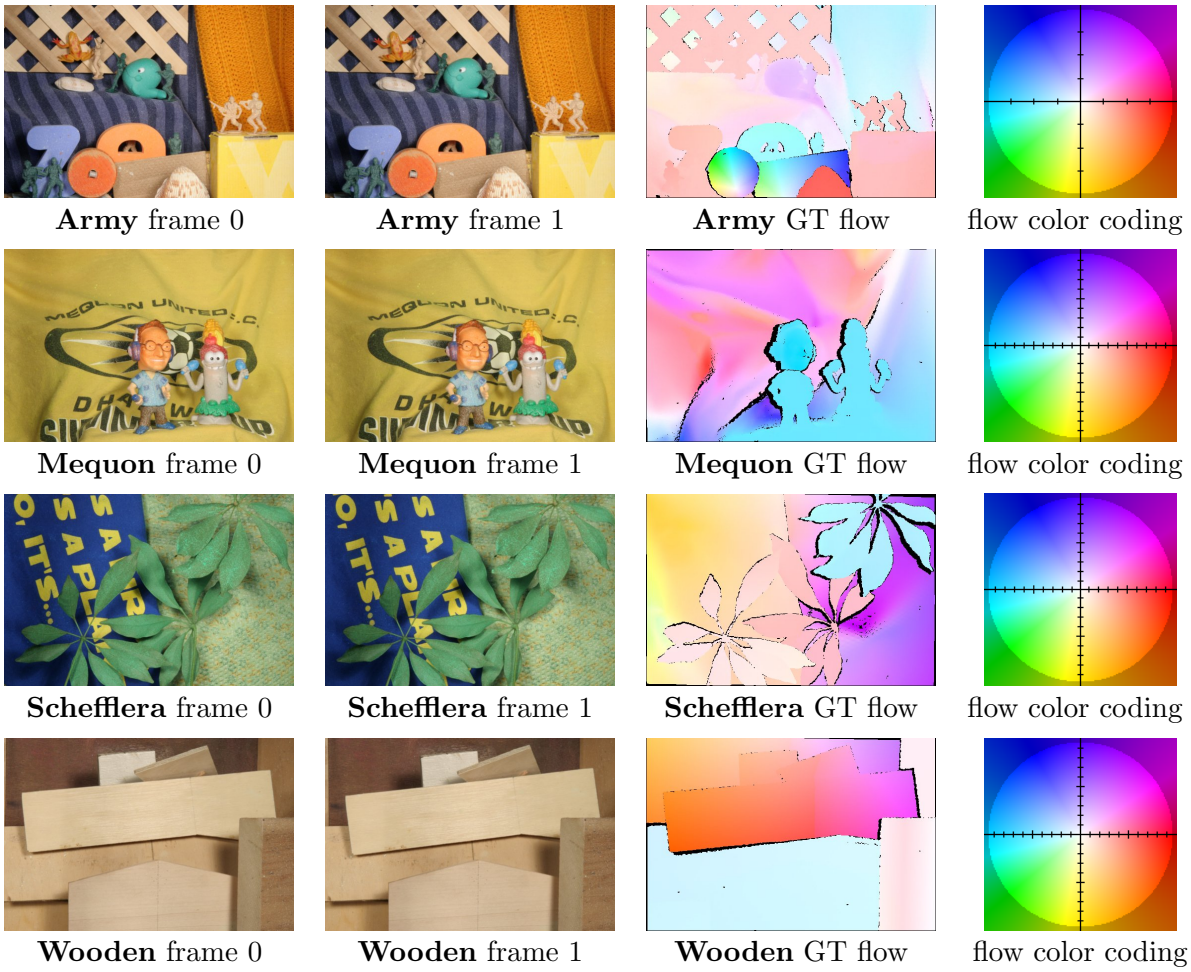


Figure 2: Hidden Texture Data. **Army** contains several independently moving objects. **Mequon** contains nonrigid motion and textureless regions. **Schefflera** contains thin structures, shadows, and foreground/background transitions with little contrast. **Wooden** contains rigidly moving objects with little texture in the presence of shadows. In the right-most column, we include a visualization of the color-coding of the optical flow. The "ticks" on the axes denote a flow unit of one pixel; note that the flow magnitudes are fairly low in **Army** (< 4 pixels), but higher in the other three scenes (up to 10 pixels).

3.2 Realistic Synthetic Imagery

Synthetic scenes generated using computer graphics are often indistinguishable from real ones. For the study of optical flow, synthetic data offers a number of benefits. In particular, it gives full control over the rendering process including material properties of the objects, while providing precise ground-truth motion and object boundaries.

To go beyond previous synthetic ground truth (e.g., the **Yosemite** sequence), we generated two types of fairly complex synthetic outdoor scenes. The first is a set of "natural" scenes (Figure 3 top) containing significant complex occlusion. These scenes consist of a random number of procedurally generated "rocks" and "trees" with randomly chosen ground texture and surface displacement. Additionally, the tree bark has significant 3D texture. The trees have a small amount of independent movement to mimic motion due to wind. The camera motions include camera rotation and 3D translation. A second set of "urban" scenes (Figure 3 middle) contain buildings generated with a random shape grammar. The "buildings" have randomly selected scanned textures and some surfaces are slightly reflective. There are cast shadows as well as a few independently moving "cars".

These scenes were generated using the 3Delight Renderman-compliant renderer [21] at a resolution of 640x480 pixels using linear gamma. The images are antialiased, mimicking the effect of sensors with finite area. Current rendered scenes do not use full global illumination but use the ambient occlusion approximation. Frames in these synthetic sequences were generated without motion blur.

The ground truth was computed using a custom shader that projects the 3D motion of the scene corresponding to a particular image onto the 2D image plane. Since individual pixels can potentially represent more than one object, simply point-sampling the flow at the center of each pixel could result in a flow vector that does not reflect the dominant motion under the pixel. On the other hand, applying antialiasing to the flow would result in an averaged flow vector at each pixel that does reflect the true motion of any object within that pixel. Instead, we clustered the flow vectors within each pixel and selected a flow vector from the dominant cluster: The flow fields are initially generated at 3× resolution, resulting in nine candidate flow vectors for each pixel. These motion vectors are grouped into two clusters using k-means. The k-means procedure is initialized with the vectors closest and furthest from the pixel's average flow as measured using the flow vector end points. The flow vector closest to the mean of the dominant cluster is then chosen to represent the flow for that pixel. The images were also generated at 3× resolution and downsampled using a bicubic filter.

We selected three synthetic sequences to include in the evaluation set (Figure 3). **Grove** contains a close-up view of a tree, with a substantial parallax and motion discontinuities. **Urban** contains images of a city, with substantial motion discontinuities, a large motion range, and an independently moving object. We also include the **Yosemite** sequence to allow some comparison with algorithms published prior to the release of our data.

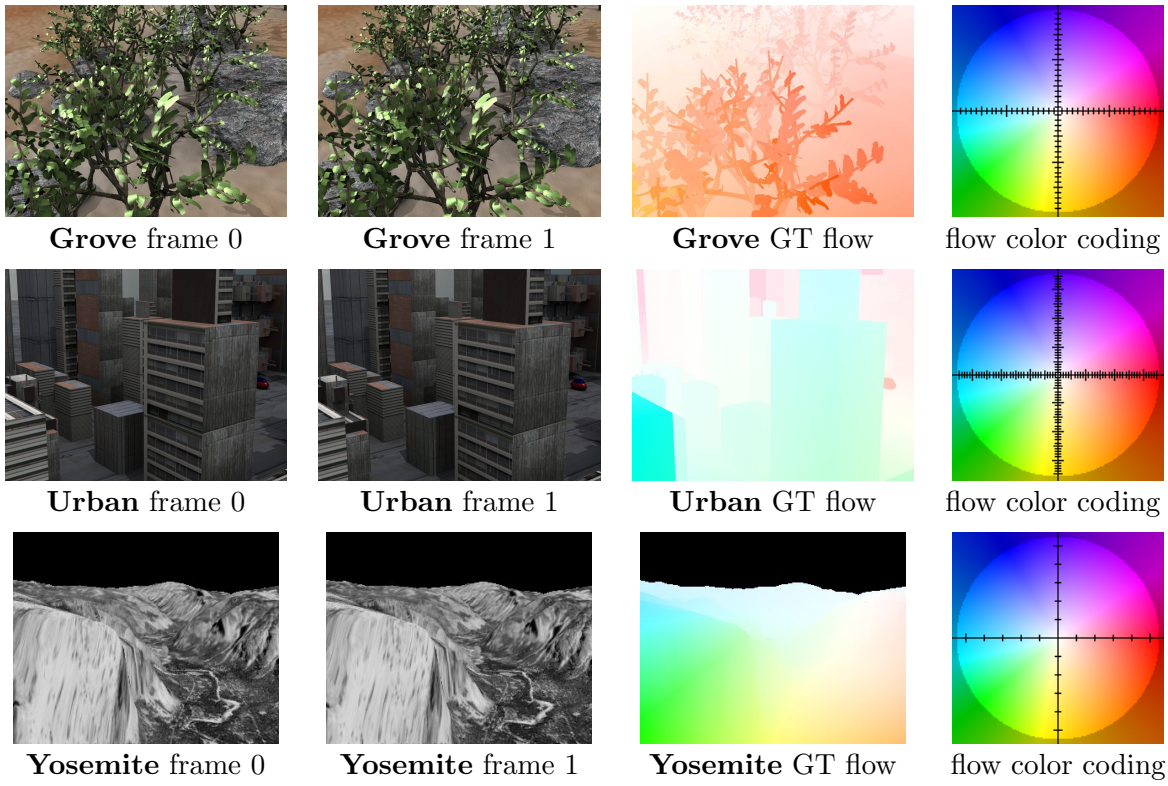


Figure 3: Synthetic Data. **Grove** contains a close up of a tree with thin structures, very complex motion discontinuities, and a large motion range (up to 20 pixels). **Urban** contains large motion discontinuities and an even larger motion range (up to 35 pixels). **Yosemite** is included in our evaluation to allow comparison with algorithms published prior to our study.

3.3 Imagery for Frame Interpolation

In a wide class of applications such as video re-timing, novel view generation, and motion-compensated compression, what is important is not how well the flow field matches the ground-truth motion, but how well intermediate frames can be predicted using the flow. To allow for measures that predict performance on such tasks, we collected a variety of data suitable for frame interpolation. The relative performance of algorithms with respect to frame interpolation and ground-truth motion estimation is interesting in its own right.

3.3.1 Frame Interpolation Datasets

We used a PointGrey Dragonfly Express camera to capture the data, acquiring 60 frames per second. We provide every other frame to the optical flow algorithms and retain the intermediate images as frame-interpolation ground truth. This temporal subsampling means that the input to the flow algorithms is captured at 30Hz while enabling generation of a 2× slow-motion sequence.

We include four such sequences in the evaluation set (Figure 4). The first two (**Backyard** and **Basketball**) include people, a common focus of many applications, but a subject matter absent from previous evaluations. **Backyard** is captured outdoors with a short shutter (6ms) and has little motion blur. **Basketball** is captured indoors with a longer shutter (16ms) and so has more motion blur. The third sequence, **Dumptruck**, is an urban scene containing several independently moving vehicles, and has substantial specularities and saturation (2ms shutter). The final sequence, **Evergreen**, includes highly textured vegetation with complex motion discontinuities (6ms shutter).

The main benefit of the interpolation dataset is that the scenes are real world scenes, captured with a real camera and containing real sources of noise. The ground truth is not a flow field, however, but an intermediate image frame. Hence, the definition of flow being used is the *apparent motion*, not the 2D projection of the motion field.

3.3.2 Frame Interpolation Algorithm

Note that the evaluation of accuracy depends on the interpolation algorithm used to construct the intermediate frame. By default, we generate the intermediate frames from the flow fields uploaded to the website using our baseline interpolation algorithm. Researchers can also upload their own interpolation results in case they want to use a more sophisticated algorithm.

Our algorithm takes a single flow field $\mathbf{u}_0$ from image I_0 to I_1 and constructs an interpolated frame I_t at time $t \in (0, 1)$. We do, however, use both frames to generate the actual intensity values. In all the experiments in this paper $t = 0.5$. Our algorithm is closely related to previous algorithms for depth-based frame interpolation [67, 85]:

1. Forward-warp the flow $\mathbf{u}_0$ to time t to give $\mathbf{u}_1$ where:

$$\mathbf{u}_t(\text{round}(\mathbf{x} + t\mathbf{u}_0(\mathbf{x}))) = \mathbf{u}_0(\mathbf{x}). \tag{19}$$

Backyard frame 0 | **Backyard** frame 1 | GT interpolated frame

Basketball frame 0 | **Basketball** frame 1 | GT interpolated frame

Dumptruck frame 0 | **Dumptruck** frame 1 | GT interpolated frame

Evergreen frame 0 | **Evergreen** frame 1 | GT interpolated frame

Figure 4: High-Speed Data for Interpolation. We collected four sequences using a PointGrey Dragonfly Express running at 60Hz. We provide every other image to the algorithms and retain the intermediate frame as interpolation ground truth. The first two sequences (**Backyard** and **Basketball**) include people, a common focus of many applications. **Dumptruck** contains several independently moving vehicles, and has substantial specularities and saturation. **Evergreen** includes highly textured vegetation with complex discontinuities.

In order to avoid sampling gaps, we splat the flow vectors with a splatting radius of ± 0.5 pixels. In cases where multiple flow vectors map to the same location, we attempt to resolve the ordering independently for each pixel by checking photoconsistency; i.e., we retain the flow $\mathbf{u}_0(\mathbf{x})$ with the lowest color difference $|I_0(\mathbf{x}) - I_1(\mathbf{x} + \mathbf{u}_0(\mathbf{x}))|$.

2. Fill any holes in $\mathbf{u}_t$ using a simple outside-in strategy.

3. Estimate occlusions masks $O_0(\mathbf{x})$ and $O_1(\mathbf{x})$, where $O_i(\mathbf{x}) = 1$ means pixel $\mathbf{x}$ in image I_i is not visible in the respective other image. To compute $O_0(\mathbf{x})$ and $O_1(\mathbf{x})$, we first forward-warp the flow $\mathbf{u}_0(\mathbf{x})$ to time $t = 1$ using the same approach as in Step 1 to give $\mathbf{u}_1(\mathbf{x})$. Any pixel $\mathbf{x}$ in $\mathbf{u}_1(\mathbf{x})$ that is not targeted by this splatting has no corresponding pixel in I_0 and thus we set $O_1(\mathbf{x}) = 1$ for all such pixels. (See [31] for a bidirectional algorithm that performs this reasoning at time t.) In order to compute $O_0(\mathbf{x})$, we cross-check the flow vectors, setting $O_0(\mathbf{x}) = 1$ if

$$|\mathbf{u}_0(\mathbf{x}) - \mathbf{u}_1(\mathbf{x} + \mathbf{u}_0(\mathbf{x}))| \;>\; 0.5. \tag{20}$$

4. Compute the colors of the interpolated pixels, taking occlusions into consideration. Let $\mathbf{x}_0 = \mathbf{x} - t\mathbf{u}_t(\mathbf{x})$ and $\mathbf{x}_1 = \mathbf{x} + (1-t)\mathbf{u}_t(\mathbf{x})$ denote the locations of the two "source" pixels in the two images. If both pixels are visible, i.e., $O_0(\mathbf{x}_0) = 0$ and $O_1(\mathbf{x}_1) = 0$, blend the two images [8]:

$$I_t(\mathbf{x}) = (1-t)I_0(\mathbf{x}_0) + tI_1(\mathbf{x}_1). \tag{21}$$

Otherwise, only sample the non-occluded image, i.e., set $I_t(\mathbf{x}) = I_0(\mathbf{x}_0)$ if $O_1(\mathbf{x}_1) = 1$ and vice versa. In order to avoid artifacts near object boundaries, we dilate the occlusion masks O_0, O_1 by a small radius before this operation. We use bilinear interpolation to sample the images.

This algorithm, while reasonable, is only meant to serve as starting point. One area for future research is to develop better frame interpolation algorithms. We hope that our database will be used both by researchers working on optical flow and on frame interpolation [31, 45].

3.4 Modified Stereo Data for Rigid Scenes

Our final type of data consists of modified stereo data. Specifically we include the **Teddy** dataset [62] in the evaluation set, the ground truth for which was obtained using structured lighting [64] (Figure 5). Stereo datasets typically have an asymmetric disparity range $[0, d_{\max}]$, which is appropriate for stereo, but not for optical flow. We crop different subregions of the images, thereby introducing a spatial shift, to convert this disparity range to $[-d_{\max}/2, d_{\max}/2]$.

A key benefit of the modified stereo dataset, like the hidden fluorescent texture dataset, is that it contains ground-truth flow fields on imagery captured with a real camera. An additional benefit is that it allows a comparison between state-of-the-art stereo algorithms

Teddy frame 0 | **Teddy** frame 1 | **Teddy** GT flow | flow color coding

Figure 5: Stereo Data. We cropped the stereo dataset **Teddy** [64] to convert the asymmetric stereo disparity range into a roughly symmetric flow field. This dataset includes complex geometric, and significant occlusions and motion discontinuities. One reason for including this dataset is to allow comparison with state-of-the-art stereo algorithms.

and optical flow algorithms. Shifting the disparity range does not affect the performance of stereo algorithms as long as they are given the new search range. Although optical flow is a more under-constrained problem, the relative performance of algorithms may lead to algorithmic insights.

One concern with the modified stereo dataset is that algorithms may take advantage of the knowledge that the motions are all horizontal. Indeed a number recent algorithms have considered rigidity priors [78, 79]. However, these algorithms must also perform well on the other types of data and any over-fitting to the rigid data should be visible by comparing results across the 12 images in the evaluation set. Another concern would be that the ground truth is only accurate to 0.25 pixels. (The original stereo data comes with pixel-accurate ground truth but is four times higher resolution [64].) The most appropriate performance statistics for this data, therefore, are the robustness statistics used in the Middlebury stereo dataset [63] (Section 4.2).

4 Evaluation Methodology

We refine and extend the evaluation methodology of [7] in terms of: (1) the performance measures used, (2) the statistics computed, and (3) the sub-regions of the images considered.

4.1 Performance Measures

The most commonly used measure of performance for optical flow is the angular error (AE). The AE between a flow vector (u, v) and the ground-truth flow $(u_{\mathrm{GT}}, v_{\mathrm{GT}})$ is the angle in 3D space between $(u, v, 1.0)$ and $(u_{\mathrm{GT}}, v_{\mathrm{GT}}, 1.0)$. The AE can be computed by taking the dot product of the vectors, dividing by the product of their lengths, and then taking the inverse

cosine:

$$\text{AE} \;=\; \arccos\left(\frac{1.0 + u \times u_{\text{GT}} + v \times v_{\text{GT}}}{\sqrt{1.0 + u \times u + v \times v}\sqrt{1.0 + u_{\text{GT}} \times u_{\text{GT}} + v_{\text{GT}} \times v_{\text{GT}}}}\right). \tag{22}$$

The popularity of this measure is based on the seminal survey by Barron et al. [7], although the measure itself dates to prior work by Fleet and Jepson [25]. The goal of the AE is to provide a *relative* measure of performance that avoids the "divide by zero" problem for zero flows. Errors in large flows are penalized less in AE than errors in small flows.

Although the AE is prevalent, it is unclear why errors in a region of smooth non-zero motion should be penalized less than errors in regions of zero motion. The AE also contains an arbitrary scaling constant (1.0) to convert the units from pixels to degrees. Hence, we also compute an *absolute* error, the error in flow endpoint (EE) used in [53] defined by:

$$\text{EE} \;=\; \sqrt{(u - u_{\text{GT}})^2 + (v - v_{\text{GT}})^2}. \tag{23}$$

Although the use of AE is common, the EE measure is probably more appropriate for most applications (see Section 5.2.1). We report both.

For image interpolation, we define the interpolation error (IE) to be the root-mean-square (RMS) difference between the ground-truth image and the estimated interpolated image

$$\text{IE} \;=\; \left[\frac{1}{N}\sum_{(x,y)} \left(I(x,y) - I_{\text{GT}}(x,y)\right)^2,\right]^{\frac{1}{2}} \tag{24}$$

where N is the number of pixels. For color images, we take the L2 norm of the vector of RGB color differences.

We also compute a second measure of interpolation performance, a gradient-normalized RMS error inspired by [72]. The normalized interpolation error (NE) between an interpolated image $I(x,y)$ and a ground-truth image $I_{\text{GT}}(x,y)$ is given by:

$$\text{NE} \;=\; \left[\frac{1}{N}\sum_{(x,y)} \frac{\left(I(x,y) - I_{\text{GT}}(x,y)\right)^2}{\|\nabla I_{\text{GT}}(x,y)\|^2 + \epsilon}\right]^{\frac{1}{2}}. \tag{25}$$

In our experiments the arbitrary scaling constant is set to be $\epsilon = 1.0$ (graylevels per pixel squared). Again, for color images, we take the L2 norm of the vector of RGB color differences and compute the gradient of each color band separately.

Naturally, an interpolation algorithm is required to generate the interpolated image from the optical flow field. In this paper, we use the baseline algorithm outlined in Section 3.3.2.

4.2 Statistics

Although the full histograms are available in a technical report, Barron et al. [7] only reports averages (AV) and standard deviations (SD). This has led most subsequent researchers to

Schefflera frame 0

Known flow vectors (**All**)

Motion discontinuities (**Disc**)

Textureless regions (**Untext**)

Figure 6: Region masks for **Schefflera**. Statistics are computed over the white pixels. **All** includes all the pixels where the ground-truth flow can be reliably determined. The **Disc** mask is computed by taking the gradient of the ground-truth flow (or pixel differencing if the ground-truth flow is unavailable), thresholding and dilating. The **Untext** regions are computed by taking the gradient of the image, thresholding and dilating.

only report these statistics. We also compute the robustness statistics used in the Middlebury stereo dataset [63]. In particular RX denotes the percentage of pixels that have an error measure above X. For the angle error (AE) we compute R2.5, R5.0, and R10.0 (degrees); for the endpoint error (EE) we compute R0.5, R1.0, and R2.0 (pixels); for the interpolation error (IE) we compute R2.5, R5.0, and R10.0 (graylevels); and for the normalized interpolation error (NE) we compute R0.5, R1.0, and R2.0 (no units). We also compute robust accuracy measures similar to those in [66]: AX denotes the accuracy of the error measure at the X$^{\text{th}}$ percentile, after sorting the errors from low to high. For the flow errors (AE and EE), we compute A50, A75, and A95. For the interpolation errors (IE and NE), we compute A90, A95, and A99.

4.3 Region Masks

It is easier to compute flow in some parts of an image than in others. For example, computing flow around motion discontinuities is hard. Computing motion in textureless regions is also hard, although interpolating in those regions should be easier. Computing statistics over such regions may highlight areas where existing algorithms are failing and spur further research in these cases. We follow the procedure in [63] and compute the error measure statistics over three types of region masks: everywhere (**All**), around motion discontinuities (**Disc**), and in textureless regions (**Untext**). We illustrate the masks for the **Schefflera** dataset in Figure 4.3.

The **All** masks for flow estimation include all the pixels where the ground-truth flow could be reliably determined. For the new synthetic sequences, this means all of the pixels. For **Yosemite**, the sky is excluded. For the hidden fluorescent texture data, pixels where cross-checking failed are excluded. Most of these pixels are around the boundary of objects, and around the boundary of the image where the pixel flows outside the second image. Similarly, for the stereo sequences, pixels where cross-checking failed are excluded [64]. Most of these pixels are pixels that are occluded in one of the images. The **All** masks for the interpolation

metrics include all of the pixels. Note that in some cases (particularly the synthetic data), the **All** masks include pixels that are visible in first image but are occluded or outside the second image. We did not remove these pixels because we believe algorithms should be able to extrapolate into these regions.

The **Disc** mask is computed by taking the gradient of the ground-truth flow field, thresholding the magnitude, and then dilating the resulting mask with a 9×9 box. If the ground-truth flow is not available, we use frame differencing to get an estimate of fast-moving regions instead. The **Untext** regions are computed by taking the gradient of the image, thresholding the magnitude, and dilating with a 3×3 box. The pixels excluded from the **All** masks are also excluded from both **Disc** and **Untext** masks.

5 Experimental Results

We now discuss our empirical findings. We start in Section 5.1 by outlining the evolution of our online evaluation since the publication of our preliminary paper [6]. In Section 5.2, we analyze the flow errors. In particular, we investigate the correlation between the various metrics, statistics, region masks, and datasets. In Section 5.3, we analyze the interpolation errors and in Section 5.4, we compare the interpolation error results with the flow error results. Finally, in Section 5.5, we compare the algorithms that have reported results using our evaluation in terms of which components of our taxonomy in Section 2 they use.

5.1 Online Evaluation

Our online evaluation at *http://vision.middlebury.edu/flow/* provides a snapshot of the state-of-the-art in optical flow. Seeded with the handful of methods that we implemented as part of our preliminary paper [6], the evaluation has quickly grown. At the time of writing, the evaluation contains results for 24 published methods and several unpublished ones. In this paper, we restrict attention to the published algorithms. Four of these methods were contributed by us (our implementations of Horn and Schunck [33], Lucas-Kanade [44], Combined Local-Global [16], and Black and Anandan [11]). Results for the 20 other methods were submitted by their authors. Of these new algorithms, two were published before 2007, 11 were published in 2008, and 7 were published in 2009.

On the evaluation website, we provide tables comparing the performance of the algorithms for each of the four error measures, i.e., endpoint error (EE), angular error (AE), interpolation error (IE), and normalized interpolation error (NE), on a set of 8 test sequences. For EE and AE, which measure flow accuracy, we use the 8 sequences for which we have ground-truth flow: **Army**, **Mequon**, **Schefflera**, **Wooden**, **Grove**, **Urban**, **Yosemite**, and **Teddy**. For IE and NE, which measure interpolation accuracy, we use only four of the above datasets (**Mequon**, **Schefflera**, **Urban**, and **Teddy**) and replace the other four with the high-speed datasets **Backyard**, **Basketball**, **Dumptruck**, and **Evergreen**. For each measure, we include a separate page for each of the eight statistics in Section 4.2. Figure 7 shows a screenshot of the first of these 32 pages, the average endpoint error (Avg. EE). For each

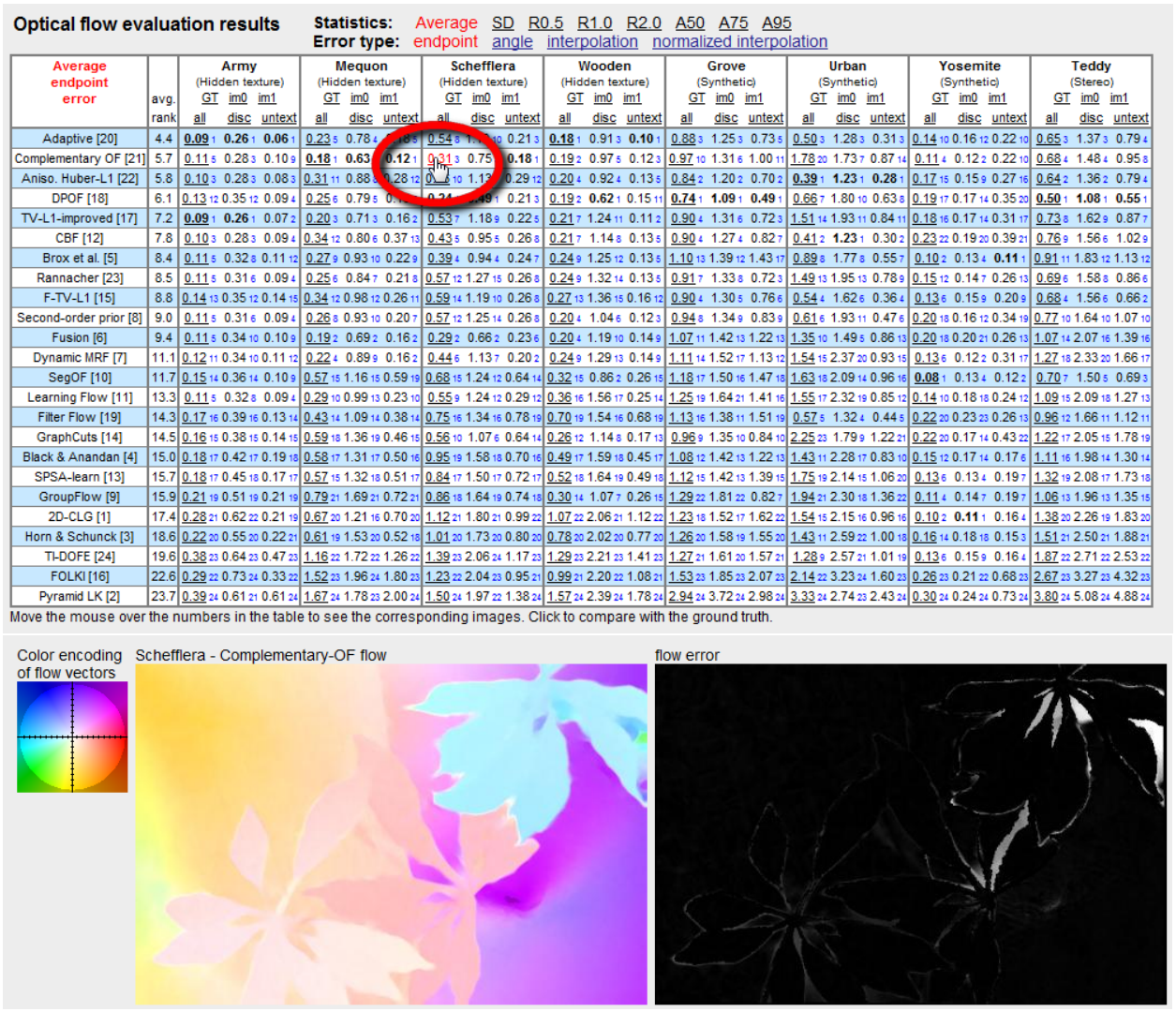


Figure 7: A screenshot of the default page at http://vision.middlebury.edu/flow/eval/, evaluating the current set of 24 published algorithms using the average endpoint error (Avg. EE). This page is one of 32 possible metric/statistic combinations the user can select. By moving the mouse pointer over an underlined performance score, the user can interactively view the corresponding flow and error maps. Clicking on a score toggles between the computed and the ground-truth flows. Next to each score, the corresponding rank in the current column is indicated with a smaller blue number. The minimum (best) score in each column is shown in boldface. The table is sorted by the average rank (computed over all 24 columns, three region masks for each of the eight sequences). The average rank serves as an *approximate* measure of performance *under the selected metric/statistic.*

Algorithm	Runtime
Adaptive [78]	9.2
Complementary OF [84]	44
Aniso. Huber-L1 [82]	2
DPOF [39]	261
TV-L1-improved [80]	2.9
CBF [74]	69
Brox et al. [54]	18
Rannacher [59]	0.12
F-TV-L1 [79]	8
Second-order prior [75]	14
Fusion [40]	2,666
Dynamic MRF [27]	366

Algorithm	Runtime
Seg OF [83]	60
Learning Flow [70]	825
Filter Flow [65]	34,000
Graph Cuts [20]	1,200
Black & Anandan [11]	328
SPSA-learn [41]	200
Group Flow [60]	600
2D-CLG [16]	844
Horn & Schunck [33]	49
TI-DOFE [19]	260
FOLKI [38]	1.4
Pyramid LK [44]	11.9

Table 1: Reported runtimes on the **Urban** sequence in seconds. We do not normalize for the programming environment, CPU speed, number of cores, or other hardware acceleration. These numbers should be treated as a very rough guideline of the inherent computational complexity of the algorithms.

measure and statistic, we evaluate all methods on the set of eight test images with three different regions masks (all, disc, and untext; see Section 4.3), resulting in a set of 24 scores per method. We sort each table by the average rank across all 24 scores to provide an ordering that *roughly* reflects the overall performance on the current metric and statistic.

We want to emphasize that we do not aim to provide an overall ranking among the submitted methods. Authors sometimes report the rank of their method on one or more of the 32 tables (often average angular error); however, many of the other 31 metric/statistic combinations might be better suited to compare the algorithms, depending on the application of interest. Also note that the exact rank within any of the tables only gives a rough measure of performance, as there are various other ways that the scores across the 24 columns could be combined.

We also list the runtimes reported by authors on the **Urban** sequence on the evaluation website (see Table 1). We made no attempt to normalize for the programming environment, CPU speed, number of cores, or other hardware acceleration. These numbers should be treated as a very rough guideline of the inherent computational complexity of the algorithms.

Finally, we report on the evaluation website for each method the number of input frames and whether color information was utilized. At the time of writing, all of the 24 published methods discussed in this paper use only 2 frames as input; and 10 of them use color information.

The best-performing algorithm (both in terms of average endpoint error and average angular error) in our preliminary study [6] was 2D-CLG [16]. In Table 2, we compare the results of 2D-CLG with the current best result in terms of average endpoint error (Avg. EE).

	Army	Mequon	Schefflera	Wooden	Grove	Urban	Yosemite	Teddy
Best	0.09	0.18	0.24	0.18	0.74	0.39	0.08	0.50
2D-CLG [16]	0.28	0.67	1.12	1.07	1.23	1.54	0.10	1.38

Table 2: A comparison of the average endpoint error (Avg. EE) results for 2D-CLG [16] (overall the best-performing algorithm in our preliminary study [6]) and the best result uploaded to the evaluation website at the time of writing (Figure 7).

The first thing to note is that performance has dramatically improved, with average EE values of less than 0.2 pixels on four of the datasets (**Yosemite**, **Army**, **Mequon**, and **Wooden**). The common elements of the more difficult sequences (**Grove**, **Teddy**, **Urban**, and **Schefflera**) are the presence of large motions and strong motion discontinuities. The complex discontinuities and fine structures of **Grove** seem to cause the most problems for current algorithms. A visual inspection of some computed flows (Figure 8) shows that oversmoothing motion discontinuities is common even for the top-performing algorithms. A possible exception is DPOF [39]. On the other hand, the problems of complex non-rigid motion confounded with illumination changes, moving shadows, and real sensor noise (**Army**, **Mequon**, **Wooden**) do not appear to present as much of a problem for current algorithms.

5.2 Analysis of the Flow Errors

We now analyze the correlation between the metrics, statistics, region masks, and datatypes for the flow errors. Figure 9 compares the average ranks computed over different subsets of the 32 pages of results, each of which contains 24 results for each algorithm. Column (a) contains the average rank computed over seven of the eight statistics (the standard deviation is omitted) and the three region masks for the endpoint error (EE). Column (b) contains the corresponding average rank for the angular error (AE). Columns (c) contain the average rank for each of the seven statistics for the endpoint error (EE) computed over the three masks and the eight datasets. Columns (d) contain the average endpoint error (Avg. EE) for each of the three masks just computed over the eight datasets. Columns (e) contains the Avg. EE computed for each of the datasets, averaged over each of the three masks. The order of the algorithms is the same as Figure 7, i.e., we order by the average endpoint error (Avg. EE), the highlighted, leftmost column in (c). To help visualize the numbers, we color-code the average ranks with a color scheme where green denotes low values, yellow intermediate, and red large values.

We also include the Pearson product-moment coefficient r between various subsets of pairs of columns at the bottom of the figure. The Pearson measure of correlation takes on values between -1.0 and 1.0, with 1.0 indicating perfect correlation. First, we include the correlation between each column and column (a). As expected, the correlation of column (a) with itself is 1.0. We also include the correlation between all pairs of the statistics, between

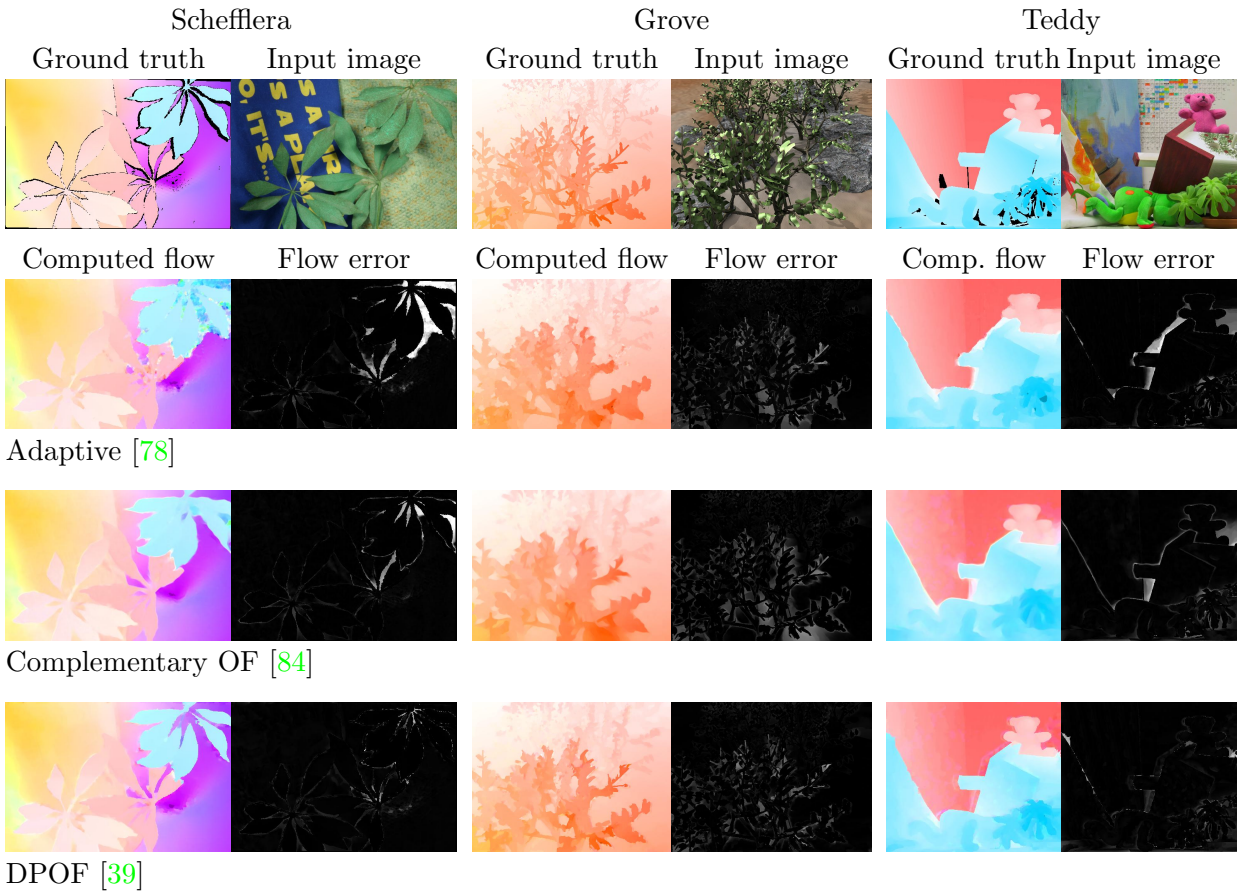


Figure 8: The results of some of the top-performing methods on three of the more difficult sequences. All three sequences contain strong motion discontinuities. **Grove** also contains particularly fine structures. The general tendency is to oversmooth motion discontinuities and fine structures. A possible exception is DPOF [39].

Flow accuracy - analysis of statistics

Method	Avg over all stats		Individual EE statistics							Avg EE by mask			Avg EE by dataset							
	EE	AE	Avg	R0.5	R1.0	R2.0	A50	A75	A95	all	disc	untext	Army	Mequ.	Scheffl.	Wood.	Grove	Urban	Yosem.	Teddy
Adaptive	4.8	4.4	4.4	5.2	5.3	4.7	3.9	3.7	6.7	4.3	4.9	4.0	1.0	4.7	7.0	1.7	3.7	3.0	10.7	3.3
Complementary OF	5.2	5.9	5.7	4.3	3.7	3.8	6.5	6.0	6.4	6.1	3.9	7.1	5.7	1.0	2.3	3.3	9.0	13.7	5.3	5.3
Aniso. Huber-L1	6.6	6.7	5.8	7.4	6.4	6.0	6.9	6.8	7.2	6.0	4.5	6.9	3.0	10.3	9.7	4.3	2.0	1.0	13.3	2.7
DPOF	6.7	8.0	6.1	7.5	5.6	4.5	9.4	7.9	6.1	5.9	5.6	6.8	9.3	5.7	1.7	4.7	1.0	8.3	17.0	1.0
TV-L1-improved	6.8	6.7	7.2	6.3	6.0	6.9	5.7	6.1	9.4	7.5	8.0	6.1	1.3	2.7	7.0	6.7	4.3	12.0	15.7	8.0
CBF	8.1	9.2	7.8	8.6	7.3	6.8	7.9	8.6	9.4	8.0	6.6	8.6	3.3	10.3	6.0	6.7	5.0	1.7	21.0	8.0
Brox et al.	8.7	8.4	8.4	9.6	9.8	7.8	7.6	8.2	9.4	7.6	8.8	8.8	8.3	9.3	5.0	8.7	14.0	7.7	2.3	11.7
Rannacher	8.1	7.5	8.5	7.2	7.8	9.1	6.4	7.2	10.8	8.8	9.8	7.0	5.0	7.0	11.7	9.3	6.0	11.7	10.7	6.7
F-TV-L1	8.3	8.1	8.8	6.8	7.9	10.5	6.8	7.5	9.5	8.8	9.4	8.4	13.3	11.7	10.7	13.3	5.0	4.7	8.0	4.0
Second-order prior	9.6	10.2	9.0	11.0	11.4	9.5	6.9	8.8	10.5	8.9	9.8	8.3	5.0	8.3	11.3	4.3	8.7	7.7	16.3	10.0
Fusion	9.3	11.8	9.4	9.1	8.7	8.2	9.9	8.8	11.0	8.3	9.9	10.1	8.0	2.0	3.3	7.7	12.3	9.3	17.3	15.3
Dynamic MRF	10.3	9.7	11.1	9.2	9.9	10.8	8.8	9.9	12.7	10.4	12.3	10.8	11.0	5.0	5.0	10.3	14.3	16.7	8.3	18.3
SegOF	12.2	12.4	11.7	14.8	13.7	9.5	12.8	13.1	10.0	12.8	10.3	12.0	12.3	16.3	13.7	10.7	17.0	16.0	2.3	5.0
Learning Flow	12.6	11.7	13.3	11.7	13.0	14.0	9.5	12.6	14.0	12.6	15.8	11.6	5.7	11.0	11.0	15.7	18.7	16.0	13.3	15.3
Filter Flow	14.2	14.2	14.3	14.5	14.4	13.2	15.2	15.5	12.6	14.8	13.9	14.3	15.3	14.0	17.0	18.0	15.3	4.7	18.7	11.3
Graph Cuts	14.5	14.5	14.5	15.6	15.4	12.0	15.6	15.2	13.5	15.5	12.0	16.1	15.0	17.3	10.0	11.0	9.7	17.7	18.7	17.0
Black & Anandan	15.5	15.7	15.0	15.2	15.8	16.7	15.2	15.6	14.8	15.1	16.0	13.8	17.3	16.7	17.7	17.3	12.7	12.7	10.7	14.7
SPSA-learn	15.0	14.9	15.7	14.8	13.8	14.6	14.8	14.5	16.5	15.8	15.1	16.1	17.3	16.7	17.0	18.3	14.3	18.0	5.7	18.0
Group Flow	16.2	16.3	15.9	17.4	18.3	15.5	16.2	16.6	13.5	16.5	15.8	15.5	19.0	21.0	18.3	12.0	17.0	20.3	6.0	13.7
2D-CLG	17.3	15.9	17.4	18.3	18.2	17.5	16.8	17.2	15.5	17.4	16.6	18.1	20.7	18.7	21.3	21.7	19.0	15.7	2.3	19.7
Horn & Schunck	18.6	19.1	18.6	18.8	19.1	19.0	18.9	19.0	16.5	18.1	20.0	17.6	20.3	19.0	20.0	20.0	19.7	17.0	11.7	21.0
TI-DOFE	19.8	20.7	19.6	21.2	20.8	19.2	20.9	20.0	17.2	18.6	20.5	19.6	23.0	22.0	23.3	23.0	20.7	16.3	6.3	22.0
FOLKI	22.2	21.8	22.6	22.1	22.5	21.8	21.5	22.3	22.6	22.4	23.1	22.4	22.7	23.3	22.0	21.3	23.0	23.0	22.7	23.0
Pyramid LK	23.2	23.1	23.7	22.8	23.3	23.4	23.2	23.3	22.9	24.0	23.1	24.0	23.0	23.7	23.3	24.0	24.0	23.7	24.0	24.0
Correlation with EE:	**1.0**	.989	.996	.985	.989	.977	.973	.993	.954	.992	.971	.986	.919	.913	.899	.920	.879	.755	.158	.870

Correlation in group:

	Avg	R0.5	R1.0	R2.0	A50	A75	A95
Avg	**1.0**	.970	.978	.981	.962	.986	.968
R0.5	.970	**1.0**	.991	.937	.972	.985	.903
R1.0	.978	.991	**1.0**	.962	.952	.980	.922
R2.0	.981	.937	.962	**1.0**	.915	.954	.964
A50	.962	.972	.952	.915	**1.0**	.986	.888
A75	.986	.985	.980	.954	.986	**1.0**	.925
A95	.968	.903	.922	.964	.888	.925	**1.0**

	all	disc	untext
all	**1.0**	.960	.985
disc	.960	**1.0**	.937
untext	.985	.937	**1.0**

	Army	Mequ.	Scheffl.	Wood.	Grove	Urban	Yosem.	Teddy
Army	**1.0**	.873	.827	.887	.783	.693	-.045	.759
Mequ.	.873	**1.0**	.909	.831	.725	.595	.041	.658
Scheffl.	.827	.909	**1.0**	.878	.741	.569	.005	.675
Wood.	.887	.831	.878	**1.0**	.831	.634	.026	.813
Grove	.783	.725	.741	.831	**1.0**	.766	-.068	.852
Urban	.693	.595	.569	.634	.766	**1.0**	-.042	.744
Yosem.	-.045	.041	.005	.026	-.068	-.042	**1.0**	.135
Teddy	.759	.658	.675	.813	.852	.744	.135	**1.0**

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

Figure 9: A comparison of the various different metrics, statistics, region masks, and datatypes for flow errors. Each column contains the average rank computed over a different subset of the 32 pages of results, each of which contains 24 different results for each algorithm. See the main body of the text for a description of exactly how each column is computed. To help visualize the numbers, we color-code the average ranks with a color scheme where green denotes low values, yellow intermediate, and red large values. The order of the algorithms is the same as Figure 7, i.e., we order by the average endpoint error (Avg. EE), the leftmost column in (c), which is highlighted in the table. At the bottom of the table, we include correlations between various subsets of pairs of the columns. Specifically, we compute the Pearson product-moment coefficient r. We separately color-code the correlations with a scale where dark green is 1.0 and yellow/red denote lower values.

(a) First input image (b) Ground-truth flow (c) Computed flow (d) Flow absolute error

Figure 10: Results of the Complementary OF algorithm [84] on the Urban sequence. The average AE is 4.64 degrees which ranks 6$^{\text{th}}$ in the table at the time of writing. The average EE is 1.78 pixels which ranks 20$^{\text{th}}$ at the time of writing. The huge discrepancy is due to the fact that the building in the bottom left has a very large motion, so the AE in that region is downweighted. Based on this example, we argue that the endpoint error (EE) should become preferred measure of flow accuracy.

all pairs of the masks, and between all pairs of the datasets. The results are shown in the 7×7, 3×3, and 8×8 (symmetric) matrices at the bottom of the table. We color-code the correlation results with a separate scale where 1.0 is dark green and yellow/red denote lower values (less correlation).

5.2.1 Comparison of the Endpoint Error and the Angular Error

Columns (a) and (b) in Figure 9 contain average ranks for the endpoint error (EE) and angular error (AE). The rankings generated with these two measures are highly correlated ($r = 0.989$), with only a few ordering reversals. At first glance, it may seem that the two measures could be used largely interchangeably. Studying the qualitative results contained in Figure 10 for the Complementary OF algorithm [84] on the Urban sequence leads to a different conclusion. The Complementary OF algorithm (which otherwise does very well) fails to correctly estimate the flow of the building in the bottom left. The average AE for this result is 4.64 degrees which ranks 6$^{\text{th}}$ in the table at the time of writing. The average EE is 1.78 pixels which ranks 20$^{\text{th}}$ at the time of writing. The huge discrepancy is due to the fact that the building in the bottom left has a very large motion, so the AE in that region is downweighted. Based on this example, we argue that the endpoint error (EE) should become the preferred measure of flow accuracy.

5.2.2 Comparison of the Statistics

Columns (c) in Figure 9 contains a comparison of the various statistics, the average (Avg), the robustness measures (R0.5, R1.0, and R2.0), and the accuracy measures (A50, A75, and A95). The first thing to note is that again these measures are all highly correlated with the average over all the statistics in column (a) and with themselves.

The outliers and variation in the measures for any one algorithm can be very informative. For example, the performance of DPOF [39] improves dramatically from R0.5 to R2.0 and

similarly from A50 to A95. This trend indicates that DPOF is good at avoiding gross outliers but is relatively weak at obtaining high accuracy. DPOF [39] is a segmentation-based discrete optimization algorithm, followed by a continuous refinement (Section 2.4.2). The variation of the results across the measures indicates that the combination of segmentation and discrete optimization is beneficial in terms of avoiding outliers, but that perhaps the continuous refinement is not as sophisticated as recent purely continuous algorithms. The qualitative results obtained by DPOF on the Schefflera and Grove sequences in Figure 8 show relatively good results around motion boundaries, supporting this conclusion.

5.2.3 Comparison of the Region Masks

Columns (d) in Figure 9 contain a comparison of the region masks, **All**, **Disc**, and **Untext**. Overall, the measures are highly correlated by rank, particularly for the **All** and **Untext** masks. When comparing the actual error scores in the individual tables (e.g., Figure 7), however, the errors are much higher throughout in the **Disc** regions than in the **All** regions, while the errors in the **Untext** regions are typically the lowest. As expected, the **Disc** regions thus capture what is still the hardest task for optical flow algorithms: to accurately recover motion boundaries. Methods that strongly smooth across motion discontinuities (such as the Horn and Schunck algorithm [33], which uses a simple L2 prior) also show a worse performance for **Disc** in the rankings (columns (d) in Figure 9). Textureless regions, on the other hand, seem to be no problem for today's methods, essentially all of which optimize a global energy.

5.2.4 Comparison of the Datasets

Columns (e) in Figure 9 contain a comparison across the datasets. The first thing to note is how relatively uncorrelated the results are. The results on the **Yosemite** sequence, in particular, are either poorly or negatively correlated with all of the others. (The main reason is that the **Yosemite** flow contains few discontinuities and consequently methods do well here that oversmooth other sequences with more motion boundaries.) The most correlated subset of results appear to be the four hidden texture sequences **Army**, **Mequon**, **Schefflera**, and **Wooden**. These results show how performance on any one sequence can be a poor predictor of performance on other sequences and how a good benchmark needs to contain as diverse a set of data as possible. Conversely, any algorithm that performs consistently well across a diverse collection of datasets can probably be expected to perform well on most inputs.

Studying the results in detail, a number of interesting conclusions can be noted. Complementary OF [84] does well on the hidden texture data (**Army**, **Mequon**, **Schefflera**, **Wooden**) presumably due to the sophisticated normalizations in the data term (the hidden texture data contains a number of moving shadows and other illumination related effects), but relatively poorly on the sequences with large motion (**Urban**) and complex discontinuities (**Grove**). DPOF [39], which involves segmentation and performs best on **Grove**, does particular poorly on **Yosemite** presumably because segmenting the grayscale **Yosemite** sequence is difficult. F-TV-L1 [79] does well on the largely rigid sequences (**Grove**, **Urban**,

Yosemite, and **Teddy**), but poorly on the non-rigid sequences (**Army**, **Mequon**, **Schefflera**, and **Wooden**). F-TV-L1 uses a rigidity prior and so it seems that this component is being used too aggressively. Note, however, that a later algorithm by the same group of researchers (Adaptive [78], which also uses a rigidity prior) appears to have addressed this problem. The flow fields for Dynamic MRF [27] all appear to be over-smoothed; however, quantitatively, the performance degradation is only apparent on the sequences with strong discontinuities (**Grove**, **Urban**, and **Teddy**). In summary, the relative performance of an algorithm across the various datatypes in our benchmark can lead to insights into which of its components work well and which are limiting performance.

5.3 Analysis of the Interpolation Errors

We now analyze the correlation between the metrics, statistics, region masks, and datatypes for the interpolation errors. In Figure 11, we include results for the interpolation errors that are analogous to the flow error results in Figure 9, described in Section 5.2. Note that we are now comparing interpolated frames (generated from the submitted flow fields using the interpolation algorithm from Section 3.3.2) with the true intermediate frames. Also, recall that we use a different set of test sequences for the interpolation evaluation: the four high-speed datasets **Backyard**, **Basketball**, **Dumptruck**, and **Evergreen**, in addition to **Mequon**, **Schefflera**, **Urban**, and **Teddy**, as representatives of the three other types of datasets. We sort the algorithms by the average interpolation error performance (Avg. IE), the leftmost column in Figure 11(c). The ordering of the algorithms in Figure 11 is therefore different from that in Figure 9.

5.3.1 Comparison of the Interpolation and Normalized Interpolation Errors

Columns (a) and (b) in Figure 11 contain average ranks for the interpolation error (IE) and the normalized interpolation error (NE). The rankings generated with these two measures are highly correlated ($r = 0.981$), with only a few ordering reversals. Most of the differences between the two measures can be explained by the relative weight given to the discontinuity and textureless regions. The rankings in columns (a) and (b) are computed by averaging the ranking over the three masks. The normalized interpolation error (NE) generally gives additional weight to textureless regions, and less weight to discontinuity regions (which often also exhibit an intensity gradient). For example, CBF [74] performs better on the **All** and **Disc** regions than it does on the **Untext** regions, which explains why the NE rank for this algorithm is slightly higher than the IE rank.

5.3.2 Comparison of the Statistics

Columns (c) in Figure 11 contain a comparison of the various statistics, the average (Avg), the robustness measures (R2.5, R5.0, and R10.0), and the accuracy measures (A90, A95, and A99). Overall the results are highly correlated. The most obvious exception is R2.5, which measures the percentage of pixels that are predicted very precisely (within 2.5 graylevels).

Interpolation accuracy - analysis of statistics

Method	Avg over all stats		Individual IE statistics							Avg IE by mask			Avg IE by dataset							
	IE	NE	Avg	R2.5	R5.0	R10	A90	A95	A99	all	disc	untext	Mequ.	Scheffl.	Urban	Teddy	Backyd.	Basktb.	Dumptr.	Evergr.
CBF	5.6	7.9	3.5	10.6	7.4	3.8	4.6	5.0	4.0	2.4	2.3	6.0	2.3	5.3	1.3	3.3	4.0	3.0	3.7	5.3
Aniso. Huber-L1	4.5	4.0	4.6	5.8	5.5	4.2	3.2	3.8	4.1	4.1	4.6	5.0	4.0	11.3	2.3	4.0	8.3	1.7	1.0	4.0
Second-order prior	5.2	4.4	5.5	4.9	5.1	6.1	4.0	5.0	5.8	5.1	6.4	4.9	3.3	8.0	6.0	3.0	6.3	4.3	3.0	9.7
Brox et al.	4.6	4.8	6.3	5.4	4.5	3.8	3.0	3.8	5.2	6.3	6.4	6.3	5.7	3.0	4.7	3.3	2.3	14.0	16.3	1.0
F-TV-L1	6.5	8.0	7.1	9.0	7.4	5.4	4.0	5.8	7.1	5.8	6.1	9.4	14.7	11.0	5.0	7.7	4.0	2.7	5.7	6.0
Filter Flow	11.7	12.8	9.7	14.8	13.1	10.8	10.5	12.0	10.7	8.9	8.6	11.5	10.7	16.0	9.0	9.3	5.3	9.7	7.0	10.3
Fusion	9.3	8.3	10.0	7.3	10.5	11.6	8.2	8.1	9.2	10.3	10.1	9.8	4.7	2.0	6.3	6.7	13.3	21.3	10.0	16.0
Black & Anandan	11.0	10.4	10.1	12.9	12.7	11.1	10.0	9.9	10.2	11.1	9.5	9.6	12.7	17.7	15.7	12.3	4.0	7.7	7.7	3.0
DPOF	9.7	10.1	10.2	11.8	10.1	8.8	8.1	9.2	10.0	9.9	12.0	8.6	15.0	1.0	15.3	6.7	13.7	9.0	8.7	12.0
2D-CLG	10.4	10.9	11.0	11.5	11.7	11.0	8.7	8.3	10.6	10.5	8.1	14.4	8.0	15.7	9.7	12.3	17.3	6.7	13.3	5.0
Horn & Schunck	13.9	13.9	11.1	17.3	16.4	14.0	13.1	13.0	12.2	12.1	10.8	10.5	9.0	20.0	13.7	16.3	4.7	5.3	13.0	7.0
Adaptive	10.1	9.3	12.5	9.1	10.2	10.2	7.7	9.6	11.4	13.1	14.9	9.5	11.7	16.7	7.0	12.0	14.3	14.3	12.0	12.0
Complementary OF	9.8	8.8	12.5	6.1	7.6	11.6	7.8	9.8	13.4	14.0	14.5	8.9	13.3	4.3	19.0	13.0	14.7	9.0	11.0	15.3
TV-L1-improved	11.0	11.6	12.8	10.8	11.3	10.9	7.8	10.7	12.5	13.1	12.9	12.4	8.3	15.3	11.0	5.0	11.7	18.3	18.3	14.3
Graph Cuts	10.2	11.5	13.0	7.1	8.0	10.9	9.0	11.0	12.2	14.1	12.9	11.9	17.0	5.3	14.0	12.0	15.7	10.3	15.7	13.7
TI-DOFE	14.7	14.4	13.5	17.4	16.2	14.4	13.8	13.9	13.4	13.4	12.9	14.4	17.3	22.3	9.0	19.0	5.0	12.7	7.3	15.7
Dynamic MRF	14.5	14.5	14.5	12.8	14.5	16.2	12.4	14.8	16.2	14.0	15.0	14.6	9.0	7.3	12.7	18.0	12.3	22.0	18.3	16.7
Learning Flow	17.3	18.1	15.8	20.1	20.0	17.6	15.8	16.0	15.7	15.6	15.6	16.3	11.0	13.3	24.0	13.7	19.3	15.3	12.0	18.0
FOLKI	19.0	20.9	15.9	22.5	22.2	18.6	19.0	18.9	16.2	14.3	13.9	19.5	20.3	22.7	15.0	20.7	11.3	16.3	10.7	10.0
Rannacher	12.6	13.2	16.0	11.5	11.9	12.5	9.0	12.0	15.0	16.9	17.5	13.8	13.0	18.0	15.3	13.0	15.7	19.0	18.3	16.0
SPSA-learn	14.5	15.6	18.0	10.7	12.6	15.2	12.4	14.9	17.8	18.8	18.4	16.9	19.0	12.3	21.7	18.3	17.7	12.3	24.0	18.7
SegOF	14.3	14.0	18.1	11.7	12.0	15.2	11.9	13.8	17.5	19.1	18.8	16.5	19.0	7.7	19.3	22.0	21.3	19.3	21.7	14.7
Group Flow	19.8	18.9	21.1	18.3	19.8	21.7	19.0	18.4	20.2	21.1	21.8	20.4	23.0	15.7	20.3	22.3	23.0	23.7	18.7	22.0
Pyramid LK	21.9	22.3	22.2	21.9	22.5	22.5	20.9	21.8	21.6	22.9	21.3	22.4	23.3	23.7	22.7	24.0	22.0	15.7	22.0	24.0
Correlation with IE:	**1.0**	.981	.916	.876	.956	.983	.987	.991	.937	.874	.835	.946	.766	.610	.796	.901	.621	.605	.592	.725

Correlation in group:

	Avg	R2.5	R5.0	R10	A90	A95	A99
Avg	**1.0**	.633	.769	.933	.859	.918	.988
R2.5	.633	**1.0**	.963	.793	.900	.847	.670
R5.0	.769	.963	**1.0**	.914	.968	.929	.799
R10	.933	.793	.914	**1.0**	.968	.973	.954
A90	.859	.900	.968	.968	**1.0**	.976	.885
A95	.918	.847	.929	.973	.976	**1.0**	.946
A99	.988	.670	.799	.954	.885	.946	**1.0**

	all	disc	untext
all	**1.0**	.979	.877
disc	.979	**1.0**	.824
untext	.877	.824	**1.0**

	Mequ.	Scheffl.	Urban	Teddy	Backyd.	Basktb.	Dumptr.	Evergr.
Mequ.	**1.0**	.422	.725	.819	.564	.395	.539	.599
Scheffl.	.422	**1.0**	.262	.549	.065	.091	.142	.173
Urban	.725	.262	**1.0**	.738	.723	.459	.658	.687
Teddy	.819	.549	.738	**1.0**	.585	.507	.611	.622
Backyd.	.564	.065	.723	.585	**1.0**	.582	.669	.759
Basktb.	.395	.091	.459	.507	.582	**1.0**	.692	.680
Dumptr.	.539	.142	.658	.611	.669	.692	**1.0**	.596
Evergr.	.599	.173	.687	.622	.759	.680	.596	**1.0**

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

Figure 11: A comparison of the various different metrics, statistics, region masks, and datatypes for interpolation errors. These results are analogous to those in Figure 9, except the results here are for interpolation errors rather than flow errors. See Section 5.2 for a description of how this table was generated. We sort the algorithms by the average interpolation error performance (Avg. IE), the first column in (c). The ordering of the algorithms is therefore different to that in Figure 9.

In regions with some texture, very accurate flow is needed to obtain the highest possible precision. Algorithms such as CBF [74] and DPOF [39], which are relatively robust but not so accurate (compare the performance of these algorithms for R0.5 and R2.0 in Figure 9), therefore perform worse in terms of R2.5 than they do in terms of R5.0 and R10.0.

5.3.3 Comparison of the Region Masks

Columns (d) in Figure 11 contain a comparison of the region masks, **All**, **Disc**, and **Untext**. The **All** and **Disc** results are highly correlated, whereas the **Untext** results are less correlated with the other two masks. Studying the detailed results on the webpage for the outliers in columns (d), there does not appear to be any obvious trend. The rankings in the **Untext** regions just appear to be somewhat more "noisy" due to the fact that for some datasets there are relatively few **Untext** pixels and all algorithms have relatively low interpolation errors in those regions. The actual error values (as opposed to their rankings) are quite different between the three regions masks. Like the flow accuracy errors (Section 5.2.3), the IE values are highest in the **Disc** regions since flow errors near object boundaries usually cause interpolation errors as well.

5.3.4 Comparison of the Datasets

Columns (e) in Figure 11 contain a comparison across the datasets. The results are relatively uncorrelated, just like the flow errors in Figure 9. The most notable outlier for interpolation is **Schefflera**. Studying the results in detail on the website, the primary cause appears to the right hand side of the images, where the plant leaves move over the textured cloth. This region is difficult for many flow algorithms because the difference in motions is small and the color difference is not great either. Only a few algorithms (e.g., DPOF [39], Fusion [40], and Dynamic MRF [27]) perform well in this region. Getting this region correct is more important in the interpolation study than in the flow error study because: (1) the background is quite highly textured, so a small flow error leads to a large interpolation error (see the error maps on the webpage) and (2) the difference between the foreground and background flows is small, so oversmoothing the foreground flow is not penalized by a huge amount in the flow errors. The algorithms that perform well in this region do not perform particularly well on the other sequences, as none of the other seven interpolation datasets contain regions with similar causes of difficulty, leading to the results being fairly uncorrelated.

5.4 Comparison of the Flow and Interpolation Errors

In Figure 12, we compare the flow errors with the interpolation errors. In the left half of the figure, we include the average rank scores, computed over all statistics (except the standard deviation) and all three masks. We compare flow endpoint errors (EE), interpolation errors (IE), and normalized interpolation errors (NE), and include two columns for each, Avg and Avg4. The first column, Avg EE, is computed over all eight flow error datasets, and corresponds exactly to column (a) in Figure 9. Similarly, the third and fifth columns, Avg IE

Correlation of flow and interpolation accuracy

Method	EE		IE		NE	
	Avg	Avg4	Avg	Avg4	Avg	Avg4
Adaptive	4.4	4.5	12.5	11.8	9.8	10.4
Complementary OF	5.7	5.6	12.5	12.4	11.0	9.3
Aniso. Huber-L1	5.8	5.9	4.6	5.4	5.0	5.1
DPOF	6.1	4.2	10.2	9.5	10.9	10.3
TV-L1-improved	7.2	7.4	12.8	9.9	12.7	9.8
CBF	7.8	6.5	3.5	3.1	5.6	4.8
Brox et al.	8.4	8.4	6.3	4.2	7.5	4.8
Rannacher	8.5	9.3	16.0	14.8	14.1	13.2
F-TV-L1	8.8	7.8	7.1	9.6	8.4	9.2
Second-order prior	9.0	9.3	5.5	5.1	5.5	5.1
Fusion	9.4	7.5	10.0	4.9	8.7	6.3
Dynamic MRF	11.1	11.3	14.5	11.8	15.3	11.3
SegOF	11.7	12.8	18.1	17.0	15.3	15.8
Learning Flow	13.3	13.3	15.8	15.5	15.2	15.6
Filter Flow	14.3	11.8	9.7	11.3	11.0	14.0
Graph Cuts	14.5	15.5	13.0	12.1	13.0	11.8
Black & Anandan	15.0	15.4	10.1	14.6	10.1	14.5
SPSA-learn	15.7	17.4	18.0	17.8	19.0	18.4
Group Flow	15.9	18.3	21.1	20.3	19.2	18.8
2D-CLG	17.4	18.8	11.0	11.4	11.6	11.3
Horn & Schunck	18.6	19.3	11.1	14.8	10.4	14.0
TI-DOFE	19.6	20.9	13.5	16.9	12.0	16.1
FOLKI	22.6	22.8	15.9	19.7	18.0	19.8
Pyramid LK	23.7	23.7	22.2	23.4	21.5	23.1

Correlation:	EE		IE		NE	
	Avg	Avg4	Avg	Avg4	Avg	Avg4
Avg EE	1.0	.983	.542	.726	.632	.793
Avg4 EE	.983	1.0	.594	.763	.663	.803
Avg IE	.542	.594	1.0	.905	.960	.873
Avg4 IE	.726	.763	.905	1.0	.895	.976
Avg NE	.632	.663	.960	.895	1.0	.902
Avg4 NE	.793	.803	.873	.976	.902	1.0

Figure 12: A comparison of the flow errors, the interpolation errors, and the normalized interpolation errors. We include two columns for the average endpoint error. The leftmost (Avg EE) is computed over all eight flow error datasets. The other column (Avg4 EE) is computed over the four sequences that are common to the flow and interpolation studies (**Mequon**, **Schefflera**, **Urban**, and **Teddy**). We also include two columns each for the average interpolation error and the average normalized interpolation error. The leftmost of each pair (Avg IE and Avg NE) are computed over all eight interpolation datasets. The other columns (Avg4 IE and Avg NE) are computed over the four sequences that are common to the flow and interpolation studies (**Mequon**, **Schefflera**, **Urban**, and **Teddy**). On the right, we include the 6×6 matrix of the correlations of the six columns on the left. As in previous figures, we separately color-code the average rank columns and the 6×6 correlation matrix.

and Avg NE, are computed over all eight interpolation error datasets, and correspond exactly to columns (a) and (b) in Figure 11. To remove any dependency on the different datasets, we provide the Avg4 columns, which are computed over the four sequences that are common to the flow and interpolation studies: **Mequon**, **Schefflera**, **Urban**, and **Teddy**.

The right half of Figure 12 shows the 6×6 matrix of the column correlations. It can be seen that the correlation between the results for Avg4 EE and Avg4 IE is only 0.763. The comparison here uses the same datasets, statistics, and masks; the only difference is the error metric, flow endpoint error (EE) vs. interpolation error (IE). Part of the reason these measures are relatively uncorrelated is that the interpolation errors are themselves a little noisy internally. As discussed above, the R2.5 and **Untext** mask results are relatively uncorrelated with the results for the other measures and masks. The main reason, however,

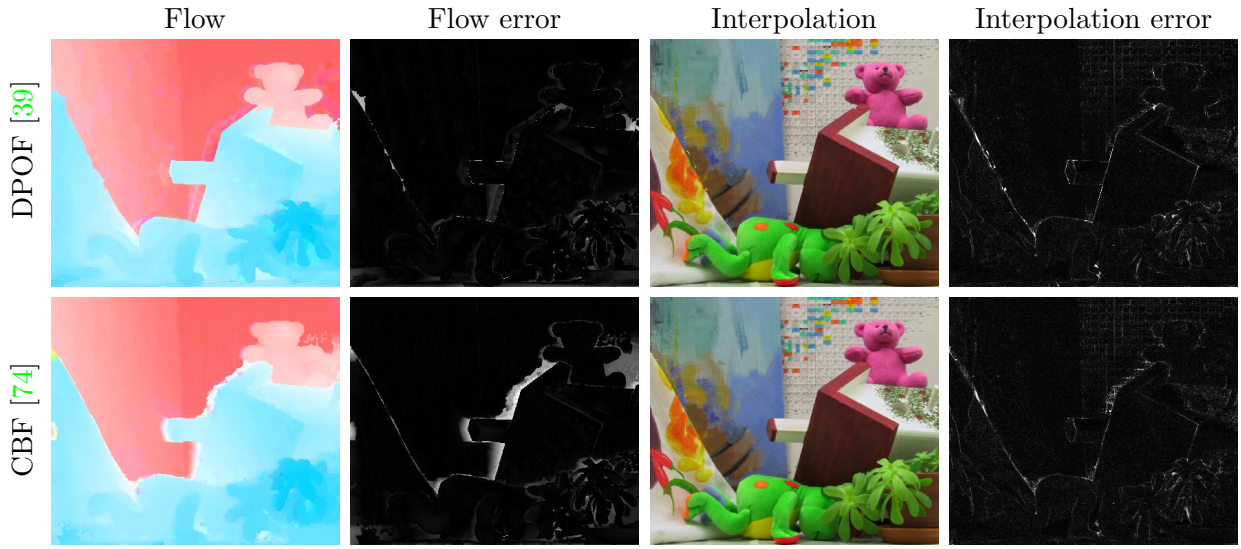


Figure 13: A comparison of the flow and interpolation results for DPOF [39] and CBF [74] on the **Teddy** sequence to illustrate the differences between the two measures of performance. DPOF obtains the best flow results with an Avg. EE of 0.5 pixels, whereas CBF is ranked 9th with an Avg. EE of 0.76 pixels. CBF obtains the best interpolation error results with an Avg. IE of 5.21 graylevels, whereas DPOF is ranked 6th with an Avg. IE of 5.58 graylevels.

is that the interpolation penalizes small flow errors in textured regions a lot, and larger flow errors in untextured regions far less. An illustration of this point is included in Figure 13. We include both flow and interpolation results for DPOF [39] and CBF [74] on the **Teddy** sequence. DPOF obtains the best flow results with an average endpoint error of 0.5 pixels, whereas CBF is the 9th best with an average endpoint error of 0.76 pixels. CBF obtains the best interpolation error results with an average interpolation error of 5.21 graylevels, whereas DPOF is 6th best with an average interpolation error of 5.58 graylevels. Although the flow errors for CBF are significantly worse, the main errors occur where the foreground flow is "fattened" into the relatively textureless background to the left of the birdhouse and the right of the teddy bear. The interpolation errors in these regions are low. On the other hand, DPOF makes flow errors on the boundary between the white cloth and blue painting that leads to large interpolation errors. The normalized interpolation error (NE) is meant to compensate for this difference between the flow and interpolation errors. Figure 12 does show that the Avg4 NE and Avg4 EE measures are more correlated ($r = 0.803$) than the Avg4 IE and Avg4 EE measures ($r = 0.763$). The increased degree of correlation is marginal, however, due to the difficulty in setting a spatial smoothing radius for the gradient computation, and the need to regularize the NE measure by adding ϵ to the denominator. Therefore, as one might expect, the performance of a method in the interpolation evaluation yields only limited information about the accuracy of the method in terms of recovering the true motion field.

	Data Term				Prior Term						Optimization					Misc.		
Algorithm	L1 Norm	Other Robust Penalty Fn	Gradient/Other Features	Illum. Modeling/Norm.	L1/TV Norm	Other Robust Penalty Fn	Spatial Weighting	Anisotropic Weighting	Higher-Order Prior	Rigidity Prior	Cont.-Gradient Descent	Cont.-Variational/Extremal	Cont.-Other	Discr.-Fusion	Discr.-Reparameterization	Learning	Visibility/Occlusion	Color
Adaptive [78]	X			X	X		X	X		X			X					
Complementary OF [84]	X		X	X		X	X	X				X						X
Aniso. Huber-L1 [82]	X			X		X	X	X					X					
DPOF [39]				X	X		X				X				X		X	X
TV-L1-improved [80]	X			X	X								X					
CBF [74]	X				X				X					X				X
Brox et al. [54]	X		X		X							X						X
F-TV-L1 [79]	X				X					X			X					
Second-order prior [75]	X								X				X					
Fusion [40]		X		X		X	X				X			X				X
Dynamic MRF [27]	X				X										X			
Seg OF [83]	X				X		X					X					X	X
Learning Flow [70]		X	X			X		X				X				X		
Filter Flow [65]	X			X	X		X		X				X					X
Graph Cuts [20]	X				X										X			X
Black & Anandan [11]		X				X					X							
SPSA-learn [41]		X				X					X					X		X
Horn & Schunck [33]												X						

Table 3: A classification of most of the algorithms for which results have been uploaded to our online evaluation in terms of which elements of our taxonomy in Section 2 they use.

5.5 Analysis of the Algorithms

Table 3 contains a summary of most of the algorithms for which results have been uploaded to our online evaluation. We omit the unpublished algorithms and a small number of the algorithms that are harder to characterize in terms of our taxonomy. We list the algorithms in the same order as Figures 7 and 9. Generally speaking, the better algorithms are at the top, although note that this is just one way to rank the algorithms. For each algorithm, we mark which elements of our taxonomy in Section 2 it uses. In terms of the data term, we mark whether the algorithm uses the L1 norm or a different robust penalty function (Section 2.1.2). Neither column is checked for an algorithm such as the Horn and Schunck [33] algorithm, which uses the L2 norm. We note if the algorithm uses a gradient component in the data term or any other more sophisticated features (Section 2.1.3). We also note if the algorithm

uses either an explicit illumination model (Section 2.1.4) or normalizes the data term in any way to reduce the effects of temporal illumination variation.

For the spatial prior term, we also mark whether the algorithm uses the Total Variation (TV) norm or a different robust penalty function (Section 2.2.2). We note if the algorithm spatially weights the prior (Section 2.2.3) or if the weighting is anisotropic (Section 2.2.4). We also note if the algorithm uses a higher-order prior (Section 2.2.5) or a rigidity prior (Section 2.2.6).

In terms of the optimization algorithm, we mark if the algorithm uses a gradient-descent based continuous optimization (Section 2.3.1). We also specify which algorithms are variational or use other extremal approaches (Section 2.3.2). Other approaches (Section 2.3.3), such as the dual variable approach and the use of Linear Programming, are grouped together. In terms of discrete optimization, we distinguish fusion based algorithms (Section 2.4.1) from reparameterization based algorithms (Section 2.4.1) and note which approaches also use a continuous optimization phase to refine the results (Section 2.4.3).

Finally, we also denote which algorithms use learning (Section 2.5.1) to optimize the parameters and which algorithms perform explicit visibility or occlusion reasoning (Section 2.5.5). In the last column we mark whether the algorithm uses color images.

Based on Table 3, we note the following:

- **Degree of Sophistication:** The algorithms toward the top of the table tend to use a lot more of the refinements to the data and prior terms. Spatial weighting, anisotropic weighting, and the addition of robustness to illumination changes through data term normalization or the use of features, are all common components in the top-performing algorithms.

- **Choice of Penalty Function:** The L1 norm is a very popular choice, particularly for the data term. A couple of the top-performing algorithms combine a L1 norm on the data term with a different (more truncated) robust penalty function on the prior term.

- **Rigidity:** As discussed in Section 5.2.4, one algorithm that uses rigidity (F-TV-L1 [79]) does poorly on the non-rigid scenes, however, Adaptive [78] (a subsequent algorithm by the same researchers) does well on all sequences.

- **Continuous Optimization:** The gradient descent algorithms (discounting the ones that first perform a discrete optimization) all appear at the bottom of the table. On the other hand, the variational approaches appear throughout the table. Note that there is a correlation between the use of variational methods and more sophisticated energy functions that is not intrinsic to the variational approach. A direct comparison of different optimization methods with the same objective functions needs to be carried out. The dual-variable approach is competitive with the best algorithms, and may offer a speed advantage.

- **Discrete Optimization:** The discrete optimization algorithms do not perform particularly well. Note, however, that the energy functions used in these methods are generally relatively simple and might be extended in the future to incorporate some of the more sophisticated elements. It does, however, appear that refining the results with a continuous optimization is required to obtain good results (if accuracy is measured using average endpoint error).

- **Miscellaneous:** There are few algorithms that employ learning in the table, making it difficult to draw conclusions in terms of performance. This is likely to change in the future, as learning techniques are maturing and more labeled training data is becoming available. Similarly, few algorithms incorporate explicit visibility or occlusion reasoning, making it difficult to assess how important this could be. Notably, all 24 algorithms considered here utilize only 2 input frames, despite the fact that we make 8-frame sequences available. In contrast, on previous evaluation sets (particularly **Yosemite**) multi-frame methods relying on temporal smoothing were quite common. This raises the question of whether temporal smoothing, at least as applied so far, is less suited for the more challenging sequences considered here. A definitive answer to this point cannot be given in this paper, but should be subject of future work. Finally, less than half of the algorithms utilize color information, and there is no obvious correlation with performance. The utility of color for image matching clearly deserves further study as well.

6 Conclusion

We have presented a collection of datasets for the evaluation of optical flow algorithms. These datasets are significantly more challenging and comprehensive than previous ones. We have also extended the set of evaluation measures and improved the evaluation methodology of Barron et al. [7]. The data and results are available at *http://vision.middlebury.edu/flow/*. Since the publication of our preliminary paper [6], a large number of authors have uploaded results to our online evaluation. The best results are a huge improvement over the algorithms in [6] (Table 2). Our data and metrics are diverse, offering a number of insights into the choice of the most appropriate metrics and statistics (Section 5.2), the effect of the datatype on the performance of algorithms and the difficulty of the various forms of data (Section 5.2.4), the differences between flow errors and interpolation errors (Section 5.3), and the importance of the various components in an algorithm (Section 5.5).

Progress on our data has been so rapid that the performance on some of the sequences is already very good (Table 2). The main exceptions are **Grove**, **Teddy**, **Urban**, and perhaps **Schefflera**. As our statistical analysis shows, however, the correlation in performance across datasets is relatively low. This suggest that no *single* method is yet able to achieve strong performance across a wide variety of datatypes. We believe that such generality is a requirement for robust optical flow algorithms suited for real-world applications.

Any such dataset and evaluation has a limited lifespan and new and more challenging

sequences should be collected. A natural question, then, is how such data is best collected. Of the various possible techniques (synthetic data [7,46], some form of hidden markers [48,58,73], human annotation [42], interpolation data [72], and modified stereo data [64]), the authors believe that synthetic data is probably the best approach (although generating high-quality synthetic data is not as easy as it might seem). Large motion discontinuities and fast motion of complex, fine structures appear to be more of a problem for current optical flow algorithms than non-rigid motion, complex illumination changes, and sensor noise. The level of difficulty is easier to control using synthetic data. Degradations such as sensor noise, etc., can also easily be added. The realism of synthetic sequences could also be improved further beyond the data in our evaluation.

Future datasets should also consider more challenging types of materials, illumination change, atmospheric effects, and transparency. Highly specular and transparent materials present not just a challenge for current algorithms, but also for quantitative evaluation. Defining the ground-truth flow and error metrics for these situations will require some care.

With any synthetic dataset, it is important to understand how representative it is of real data. Hence, the use of multiple types of data and an analysis of the correlation across them is critical. A diverse set of datatypes also reduces overfitting to any one type, while offering insights into the relative performance of the algorithms in different scenarios. On balance, however, we would recommend that any future studies contain a higher proportion of challenging, realistic synthetic data. Future studies should also extend the data to longer sequences than the 8-frame sequences that we collected.

Acknowledgments

Many thanks to Brad Hiebert-Treuer and Alan Lim for their help in creating the fluorescent texture data sets. Michael Black and Stefan Roth were supported by NSF grants IIS-0535075 and IIS-0534858, and a gift from Intel Corporation. Daniel Scharstein was supported by NSF grant IIS-0413169. Aghiles Kheffache generously donated a software license for the 3Delight renderer for use on this project. Michael Black and JP Lewis thank Lance Williams for early discussions on synthetic flow databases and Doug Creel and Luca Fascione for discussions of rendering issues. Thanks to Sing Bing Kang, Simon Winder, and Larry Zitnick for providing implementations of various algorithms. Finally, thanks to all the authors who have used our data and uploaded results to our website.

References

[1] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):384–401, 1985.

[2] J. Aggarwal and N. Nandhakumar. On the computation of motion from sequences of images—A review. *Proceedings of the IEEE*, 76(8):917–935, 1988.

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

[4] P. Anandan and R. Weiss. Introducing smoothness constraint in a matching approach for the computation of displacement fields. In *Proceedings of the DARPA Image Understanding Workshop*, pages 186–196, 1985.

[5] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 46(3):221–255, 2004.

[6] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *Proceedings of the IEEE International Conference on Computer Vision*, 2007.

[7] J. Barron, D. Fleet, and S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[8] T. Beier and S. Neely. Feature-based image metamorphosis. *ACM Computer Graphics, Annual Conference Series (SIGGRAPH)*, 26(2):35–42, 1992.

[9] J. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proceedings of the Second European Conference on Computer Vision*, pages 237–252, 1992.

[10] M. Black and P. Anandan. Robust dynamic motion estimation over time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 296–302, 1991.

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

[12] M. Black and A. Jepson. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):972–986, 1996.

[13] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, 1987.

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

[15] T. Brox, C. Bregler, and J. Malik. Large displacement optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[16] A. Bruhn, J. Weickert, and C. Schnorr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[17] P. Burt, C. Yen, and X. Xu. Local correlation measures for motion analysis: A comparative study. *IEEE Proc. PRIP*, pages 269–274, 1982.

[18] P. Burt, C. Yen, and X. Xu. Multi-resolution flow-through motion analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 246–252, 1983.

[19] C. Cassisa, S. Simoens, and V. Prinet. Two-frame optical flow formulation in an unwarped multiresolution scheme. In *Proceedings of the Iberoamerican Congress on Pattern Recognition*, pages 790–797, 2009.

[20] T. Cooke. Two applications of graph-cuts to image processing. In *Digital Image Computing: Techniques and Applications*, pages 498–504, 2008.

[21] DNA Research. 3Delight rendering software. *http://www.3delight.com/*.

[22] W. Enkelman. Investigations of multigrid algorithms for the estimation of optical flow fields in image sequences. In *Proceedings of the Workshop on Motion: Representations and Analysis*, pages 81–87, 1986.

[23] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes challenge 2009. http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html.

[24] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[25] D. Fleet and A. Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5(1):77–104, 1990.

[26] F. Glazer, G. Reyonds, and P. Anandan. Scene matching by hierarchical correlation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 432–441, 1983.

[27] B. Glocker, N. Paragios, N. Komodakis, G. Tziritas, and N. Navab. Optical flow estimation with uncertainties through dynamic MRFs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[28] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-PIE. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, 2008.

[29] K. Hanna. Direct multi-resolution estimation of ego-motion and stucture from motion. In *Proceedings of the IEEE Workshop on Visual Motion*, pages 156–162, 1991.

[30] H. Haussecker and D. Fleet. Computing optical flow with physical models of brightness variation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 760–767, 2000.

[31] E. Herbst, S. Seitz, and S. Baker. Occlusion reasoning for temporal interpolation using optical flow. Technical Report UW-CSE-09-08-01, University of Washington, Department of Computer Science and Engineering, 2009.

[32] B. Horn. *Robot Vision*. MIT Press, Cambridge, Massachusetts, 1986.

[33] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[34] A. Jepson and M. Black. Mixture models for optical flow computation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 760–761, 1993.

[35] S. Ju. *Estimating Image Motion in Layers: The Skin and Bones Model*. PhD thesis, Department of Computer Science, University of Toronto, 1998.

[36] S. Ju, M. Black, and A. Jepson. Skin and bones: Multi-layer, locally affine, optical flow and regularization of transparency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 307–314, 1996.

[37] H. Jung, K. Lee, and S. Lee. Toward global minimum through combined local minima. In *Proceedings of the European Conference on Computer Vision*, volume 4, pages 298–311, 2008.

[38] G. Le Besnerais and F. Champagnat. Dense optical flow by iterative local window registration. In *Proceedings of the International Conference on Image Processing*, volume 1, pages 137–140, 2005.

[39] C. Lei and Y. Yang. Optical flow estimation on coarse-to-fine region-trees using discrete optimization. In *Proceedings of the IEEE International Conference on Computer Vision*, 2009.

[40] V. Lempitsky, S. Roth, and C. Rother. Fusion flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[41] Y. Li and D. Huttenlocher. Learning for optical flow using stochastic optimization. In *Proceedings of the European Conference on Computer Vision*, volume 2, pages 373–391, 2008.

[42] C. Liu, W. Freeman, E. Adelson, and Y. Weiss. Human-assisted motion annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[43] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. Freeman. SIFT flow: Dense correspondence across difference scenes. In *Proceedings of the European Conference on Computer Vision*, volume 3, pages 28–42, 2008.

[44] B. Lucas and T. Kanade. An iterative image registration technique with an application in stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[45] D. Mahajan, F. Huang, W. Matusik, R. Ramamoorthi, and P. Belhumeur. Moving gradients: A path-based method for plausible image interpolation. In *ACM Computer Graphics, Annual Conference Series (SIGGRAPH)*, 2009.

[46] B. McCane, K. Novins, D. Crannitch, and B. Galvin. On benchmarking optical flow. *Computer Vision and Image Understanding*, 84(1):126–143, 2001.

[47] A. Mitiche and P. Bouthemy. Computation and analysis of image motion: A synopsis of current problems and methods. *International Journal of Computer Vision*, 19(1):29–55, 1996.

[48] Mova LLC. Contour reality capture. *http://www.mova.com/*.

[49] D. Murray and B. Buxton. Scene segmentation from visual motion using global optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2):220–228, 1987.

[50] H.-H. Nagel and W. Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5):565–593, 1986.

[51] S. Negahdaripour. Revised definition of optical flow: Integration of radiometric and geometric cues for dynamic scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(9):961–979, 1998.

[52] T. Nir, A. Bruckstein, and R. Kimmel. Over-parameterized variational optical flow. *International Journal of Computer Vision*, 76(2):205–216, February 2008.

[53] M. Otte and H.-H. Nagel. Optical flow estimation: Advances and comparisons. In *Proceedings of the European Conference on Computer Vision*, pages 51–60, 1994.

[54] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert. Highly accurate optic flow computation with theoretically justified warping. *International Journal of Computer Vision*, 67(2):141–158, 2006.

[55] P. Philips, W. Scruggs, A. O'Toole, P. Flynn, K. Bowyer, C. Schott, and M. Sharpe. Overview of the face recognition grand challenge. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 947–954, 2005.

[56] T. Pock, M. Pock, and H. Bischof. Algorithmic differentiation: Application to variational problems in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1180–1193, 2007.

[57] W. Pratt. Correlation techniques of image registration. *IEEE Trans. Aerospace and Electronic Systems AES-10*, pages 353–358, 1974.

[58] K. Ramnath, S. Baker, I. Matthews, and S. Baker. Increasing the density of active appearance models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

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

[60] X. Ren. Local grouping for optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

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

[62] D. Scharstein and C. Pal. Learning conditional random fields for stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[63] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1–3):7–42, 2002.

[64] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 195–202, 2003.

[65] S. Seitz and S. Baker. Filter flow. In *Proceedings of the IEEE International Conference on Computer Vision*, 2009.

[66] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 519–526, 2006.

[67] J. Shade, S. Gortler, L.-W. He, and R. Szeliski. Layered depth images. In *ACM Computer Graphics, Annual Conference Series (SIGGRAPH)*, pages 231–242, 1998.

[68] T. Sim, S. Baker, and M. Bsat. The CMU pose, illumination, and expression database. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1615–1618, 2003.

[69] C. Stiller and J. Konrad. Estimating motion in image sequences: A tutorial on modeling and computation of 2D motion. *IEEE Signal Processing Magazine*, 16(4):70–91, 1999.

[70] D. Sun, S. Roth, J. Lewis, and M. Black. Learning optical flow. In *Proceedings of the European Conference on Computer Vision*, volume 3, pages 83–97, 2008.

[71] J. Sun, H.-Y. Shum, and N. Zheng. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003.

[72] R. Szeliski. Prediction error as a quality metric for motion and stereo. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 781–788, 1999.

[73] M. Tappen, E. Adelson, and W. Freeman. Estimating intrinsic component images using non-linear regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1992–1999, 2006.

[74] W. Trobin, T. Pock, D. Cremers, and H. Bischof. Continuous energy minimization via repeated binary fusion. In *Proceedings of the European Conference on Computer Vision*, volume 4, pages 677–690, 2008.

[75] W. Trobin, T. Pock, D. Cremers, and H. Bischof. An unbiased second-order prior for high-accuracy motion estimation. In *Proceedings of Pattern Recognition (DAGM)*, pages 396–405, 2008.

[76] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):475–480, 2005.

[77] J. Wang and E. Adelson. Layered representation for motion analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 361–366, 1993.

[78] A. Wedel, D. Cremers, T. Pock, and H. Bischof. Structure- and motion-adaptive regularization for high accuracy optic flow. In *Proceedings of the IEEE International Conference on Computer Vision*, 2009.

[79] A. Wedel, T. Pock, J. Braun, U. Franke, and D. Cremers. Duality TV-L1 flow with fundamental matrix prior. In *Proceedings of Image and Vision Computing New Zealand*, 2008.

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

[81] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 520–526, 1997.

[82] M. Werlberger, W. Trobin, T. Pock, H. Bischof, A. Wedel, and D. Cremers. Anisotropic Huber-L1 optical flow. In *Proceedings of the British Machine Vision Conference*, 2009.

[83] L. Xu, J. Chen, and J. Jia. A segmentation based variational model for accurate optical flow estimation. In *Proceedings of the European Conference on Computer Vision*, volume 1, pages 671–684, 2008.

[84] H. Zimmer, A. Bruhn, J. Weickert, L. Valgaerts, A. Salgado, B. Rosenhahn, and H.-P. Seidel. Complementary optic flow. In *Proceedings of Seventh International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2009.

[85] C. Zitnick, S. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. *ACM Transactions on Graphics*, 23(3):600–608, 2004.