Optical Flow with Geometric Occlusion Estimation and Fusion of Multiple Frames

Ryan Kennedy and Camillo J. Taylor

Department of Computer and Information Science
University of Pennsylvania
{kenry,cjtaylor}@cis.upenn.edu

Abstract. Optical flow research has made significant progress in recent years and it can now be computed efficiently and accurately for many images. However, complex motions, large displacements, and difficult imaging conditions are still problematic. In this paper, we present a framework for estimating optical flow which leads to improvements on these difficult cases by 1) estimating occlusions and 2) using additional temporal information. First, we divide the image into discrete triangles and show how this allows for occluded regions to be naturally estimated and directly incorporated into the optimization algorithm. We additionally propose a novel method of dealing with temporal information in image sequences by using "inertial estimates" of the flow. These estimates are combined using a classifier-based fusion scheme, which significantly improves results. These contributions are evaluated on three different optical flow datasets, and we achieve state-of-the-art results on MPI-Sintel.

1 Introduction

Optical flow has a long history and many different methods have been used. Several modern methods have their roots in the seminal work of Horn and Schunck [1]. Current variants of this approach employ robust cost functions [2] and modern optimization techniques [3] and can compute optical flow accurately and efficiently for many types of images. This is corroborated by results on the Middlebury dataset [4], for which the top-performing algorithms are nearly error-free.

Despite this success, optical flow is far from solved. The underlying assumption of most models is that matching pixels should have similar intensity values and nearby pixels should have similar motions. However, these assumptions are violated in many situations, especially when motion blur, lighting variation, large motions, and atmospheric effects are involved. While Middlebury does not contain these real-world situations, more recent datasets [5,6] have many of these difficulties and the error rates of the best methods are correspondingly much higher. It remains an open question of how best to deal with these complex motions and imaging conditions.

In this paper, we present a framework for optical flow that leads to an improvement for difficult datasets. Our method is based on a triangulation of the image domain, over which we compute optical flow. We employ a tectonic model where the triangular facets are allowed to move relative to each other and a numerical quadrature scheme is used to handle the resulting occlusion effects. This allows for occlusions to be directly incorporated into the optimization procedure without the need for arbitrary regularization terms.

Additionally, we describe a novel way to incorporate temporal information over multiple frames. First, we propose "inertial estimates," which are estimates of the optical flow based on nearby frames. Next, we combine these estimates using a classifier-based fusion. Although fusion-based methods have been used previously [7], our approach is fundamentally different and does not require optimizing an NP-hard quadratic binary program. The approach is also agnostic to the underlying optical flow method and could be used with other algorithms.

In summary, we make the following four contributions:

– We show how occluded regions can be easily detected during optimization by using a triangulation of the image domain.
– We introduce "inertial estimates" which provide multiple motion estimates based on nearby frames.
– We show how fusing inertial estimates using a classifier provides a simple and effective means of incorporating temporal information into optical flow.
– We evaluate our method on multiple datasets, achieving state-of-the-art accuracy on the difficult MPI-Sintel dataset [5].

2 Related Work

Occlusions are often modeled as noise and handled through a robust cost function [2], but an explicit model for occlusions is desirable in many situations. One common approach is to compare forward and backward flow estimates, which will not match for occluded pixels [8]. Strecha *et al.* [9] use a probabilistic framework and estimate occlusions as latent variables. Occlusions can also be dealt with by incorporating layers into the model [10]. In [11], layer ordering was determined using the relative data cost between overlapping layers. Another approach is that of Xu *et al.* [12] and Kim *et al.* [13], who label points as occluded when multiple pixels map to a single point. The most similar approach to our own is from Sun *et al.* [14], who jointly estimate motion and occlusions, although their method is significantly more complex than our own and is only used to fine-tune flow fields computed using other methods.

Multi-frame optical flow estimation is often approached by assuming that flow estimates are smooth over time as well as space [15,10,16], rather than using a fusion method as we do here. Fusion methods have usually been proposed in the context of fusing the results of multiple algorithms [17] rather than incorporating temporal information. Lempitsky *et al.* [7] write the fusion problem as a quadratic binary program which they then approximate the solution to, while Jung *et al.* [18] use an algorithm involving random partitions of the flow estimates.

Triangulations of an image were used in optical flow estimation by Glocker, *et al.* [11], although otherwise their approach is quite different from ours; their triangulations are used for incorporating higher-order likelihood terms, while we use triangulations to model the flow and the resulting occlusions. Superpixel-based approaches that are not based on triangles have also been previously applied to optical flow problems [19].

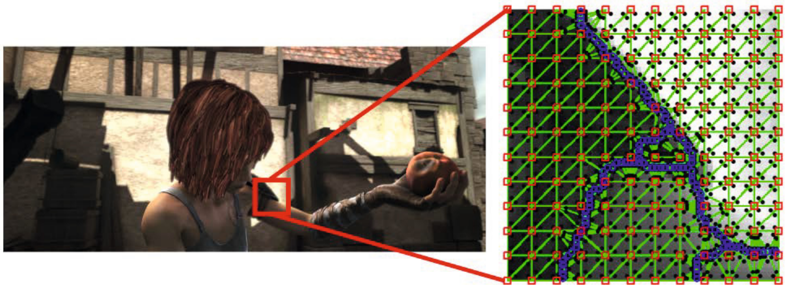

Fig. 1. A triangulated section of an image. Blue circles denote edge points and red squares denote points generated on a uniform grid with a spacing of 5 pixels. The Delaunay triangulation given by the green lines tessellates the image into regions which form the basis of our algorithm. In practice, the data cost functions are evaluated at a set of quadrature points within each triangle, shown here as black dots.

3 Problem Setup

Let $I_1, I_2 : (\Omega \subseteq \mathbb{R}^2) \to \mathbb{R}^d$ be two d-dimensional images. In this paper, we consider both to be color images in the CIELab color space such that $d = 3$. Channels are denoted using a superscript, such as $I_1^{(c)}$. We attempt to estimate the motion of each point from I_1 to I_2. The estimated motions in the horizontal and vertical directions are denoted by u and v respectively. Let $\mathbf{x} = (x, y)$ be a point in Ω, and let $f : \Omega \to \mathbb{R}^2$ be a function such that $f(\mathbf{x}) = (u(\mathbf{x}), v(\mathbf{x}))$, that is f returns the estimated motion vector associated with every point in the image. In addition, we estimate a function $m : \Omega \to \mathbb{R}$ which is a multiplicative factor that measures changes in lightness between frames, as we will define in Section 4. This "generalized brightness constancy" model has been previously used [20], and we found that it improved our results.

A key aspect of our approach is that we consider the image to be a continuous 2D function of the image domain, rather than a set of sampled pixel locations. We do so by extending the sampled pixel values to intermediate locations in the image plane using bicubic interpolation. More specifically, at any continuous-valued location $(x, y) \in \Omega$, the value of the image at channel c is computed using a quadratic form

$$I_1^{(c)}(x, y) = \left[x^3 \; x^2 \; x \; 1\right] K_c \left[y^3 \; y^2 \; y \; 1\right]^T , \tag{1}$$

where K_c is the matrix of coefficients based on the channel values of nearby pixels. Note that spatial image derivatives at any point are easily computed using derivatives of this quadratic form. Given this representation of the image as a continuous function, our goal is to compute a corresponding continuous function $f(\cdot)$ that specifies the motion of each point in Ω, along with a multiplicative brightness factor $m(\cdot)$.

We discretize the problem by tessellating the image I_1 into discrete triangular regions (Figure 1), and then seek to estimate a constant motion vector $f(\cdot)$ and brightness offset $m(\cdot)$ for each triangle. Because we assume the motion to be constant within each triangle, the triangles should be made to conform to the content of the image in order to find an accurate solution. This approach is similar to that of [11], where a triangulation of the image domain was also used. We use the following procedure. First, we extract edges from the image I_1 by using the method of [21] and threshold the given ultrametric contour map at 0.2. Each edge pixel in the image is then used as a vertex in our triangulation. In addition to these points, we also use a set of grid points that are evenly spaced throughout the 2D image, which serve to limit the maximum dimension of the resulting triangles. The grid points and edge pixels are combined and a Delaunay triangulation is constructed. An example of a tessellated image is shown in Figure 1.

4 Cost Function

Our cost function consists of data terms and smoothness terms. The data terms penalize incorrectly-matched pixels based on image data, while the smoothness terms encourage solutions that are smooth over the image domain. Our cost function takes the form

$$\mathcal{E}(f,m) = \mathcal{D}(f,m) + \tau_0\mathcal{F}(f) + \tau_1\mathcal{S}_1(f) + \tau_2\mathcal{S}_2(f) + \tau_3\mathcal{S}_3(m)\ , \qquad (2)$$

where $\mathcal{D}(\cdot)$ is a data cost term based on image data, $\mathcal{F}(\cdot)$ is a feature matching term, and $\mathcal{S}_1(\cdot), \mathcal{S}_2(\cdot)$ and $\mathcal{S}_3(\cdot)$ are smoothness terms. The parameters τ_0, τ_1, τ_2 and τ_3 control the tradeoff between these terms.

The cost function will be defined as an integral over the entire continuous image domain. We approximate this continuous integral by considering a discrete set of *quadrature points* within each triangle using the scheme described in [22]. The integral is then approximated by forming a weighted sum of the cost function evaluated at these points. We used 3 quadrature points per triangle, as shown in Figure 1.

4.1 Data Term

Our data term is given by the equation

$$\mathcal{D}(f,m) = \int_{\Omega} \Phi_\gamma \left(I_2(\mathbf{x} + f(\mathbf{x})) - \begin{bmatrix} m(\mathbf{x}) & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} I_1(\mathbf{x}) \right) d\mathbf{x}\ , \qquad (3)$$

where $\Phi_\gamma(\cdot)$ is a robust error function with parameter vector γ. Because of the large amount of data made available in the MPI-Sintel dataset, we chose our robust cost function through a fitting procedure. In particular, the difference values, $I_2^{(c)}(\mathbf{x} + f(\mathbf{x})) - I_1^{(c)}(\mathbf{x})$, are well-modeled by a Cauchy distribution, as has been previously observed [23]. The robust function $\Phi_\gamma(\cdot)$ is then the negative log-likelihood of the Cauchy density function, summed over all channels:

$$\Phi_\gamma(\delta) = \sum_{c=1}^{d} \log\left[\pi(\delta_c^2 + \gamma_c^2)/\gamma_c\right]\ . \qquad (4)$$

A separate distribution was fit to the lightness and to the combined color channels, giving values of $\gamma_1 = 0.3044$ for lightness and $\gamma_2 = \gamma_3 = 0.2012$ for the color channels.

4.2 Feature Matching Term

Feature matching has been shown to be effective at improving optical flow results, especially for large motions [24,12]. We use HOG features [25], computed *densely* at every pixel. These descriptors are then matched to their nearest neighbor in the opposite image using the approximate nearest neighbors library FLANN [26]. The matches from I_1 to I_2 generate motion estimates for each of the pixels, which we denote as $f_{HOG} : \Omega \to \mathbb{R}^2$.

If the HOG match is correct, then it is desirable to have $f(\mathbf{x})$ be close to $f_{HOG}(\mathbf{x})$. Thus, our feature matching term is given by

$$\mathcal{F}(f) = \int_{\Omega} s(\mathbf{x}) \Psi_\alpha \left(\|f(\mathbf{x}) - f_{HOG}(\mathbf{x})\|_2 \right) \, d\mathbf{x} \, , \tag{5}$$

where

$$\Psi_\alpha(\delta) = (\delta^2 + \epsilon)^\alpha \tag{6}$$

is a robust cost function with parameter α and small constant epsilon (i.e., $\epsilon = 0.001$) [3]. For $\alpha = 1$, this is a pseudo-ℓ_2 penalty. As α decreases, it becomes less convex with it becoming a pseudo-ℓ_1 penalty for $\alpha = 0.5$. For our feature matching term, we set $\alpha = 0.5$.

The function $s : \Omega \to \mathbb{R}$ is a weighting function which measures the confidence in each HOG match, and is defined as follows. First, we enforce forward-backward consistency by setting $s(\mathbf{x}) = 0$ if a match is not a mutual nearest-neighbor. Otherwise, we let $s(\mathbf{x}) = \left((d_2 - d_1)/d_1\right)^{0.2}$, where d_i is the ℓ_1 distance between the HOG feature vector in I_1 at location $\mathbf{x}$ and its i^{th}-closest match in I_2. This is similar to the weight used in [24] and provides a measure of confidence for each HOG match.

When evaluating this term on a triangulation, each triangle is assigned a HOG flow estimate by taking the mean of all flow values within the triangle t weighted by their confidence scores, $\sum_{\mathbf{x} \in t} \frac{s(\mathbf{x})}{\sum_{\mathbf{x} \in t} s(\mathbf{x})} f_{HOG}(\mathbf{x})$. The confidence of each triangle is similarly set to the average of its confidence values. These flow values are then used for all quadrature points within each triangle when evaluating the cost function.

While we used HOG features due to their speed and simplicity, more complex feature matching could be used here as well, such as [27] or [28].

4.3 Smoothness Terms

We use two different smoothness terms in our cost function: a first-order term that penalizes non-constant flow fields, and a second-order term that penalizes non-affine flow fields.

First-Order Smoothness. A first-order smoothness term penalizes non-constant motion estimates. In our cost function, all pairs of neighboring triangles are considered. The cost is defined as

$$\mathcal{S}_1(f) = \sum_{t_i, t_j \in N} |t_i||t_j| \Psi_\alpha \left(\frac{\|f(t_i) - f(t_j)\|_2}{\|\bar{t_i} - \bar{t_j}\|_2} \right) \, , \tag{7}$$

where $N \subseteq T \times T$ is the set of all neighboring triangles T in the tessellation, $\bar{t_i}$ is the centroid of triangle $t_i \in T$, and $|t_i|$ is its area. The function $\Psi_\alpha(\cdot)$ is a robust cost function, which was defined in Equation (6).

This cost function penalizes differences in the flows between neighboring triangles, modulated by the distance between their centroids. Note that we also multiply by the area of the two triangles (rather than by the edge length), which effectively connects all points within one triangle to all points in the other triangle. Now, recall that our triangulation is constructed using both edges points and a set of uniform grid points (Figure 1). The triangles along edges will therefore tend to have a smaller area, resulting in a weaker smoothness constraint. In this way, our triangulation naturally allows for a non-local smoothness cost [29].

We also apply this same smoothness cost to the multiplicative term m to encourage only locally-consistent changes in image brightness. This is denoted as the function $\mathcal{S}_3(m)$, and for this we use $\alpha = 0.5$.

Second-Order Smoothness. While a first-order smoothness term penalizes non-constant flows, a second-order smoothness term penalizes non-planar flows. This allows for motion fields with a constant gradient, which is important for datasets where such motions are common, such as KITTI (Section 8.3).

Intuitively, our second order smoothness term says that the flow of each triangle is encouraged to be near the plane that is formed from the flow values of its three neighbors. Formally, the cost function is written as a sum of costs over all triangles $t \in T$:

$$\mathcal{S}_2(f) = \sum_{t \in T} |t_i||t_j||t_k| \Psi_\alpha \left(\frac{\| f(t) - [\lambda_i f(t_i) + \lambda_j f(t_j) + \lambda_k f(t_k)] \|_2}{|\Delta_{ijk}|} \right) . \quad (8)$$

Here, t_i, t_j and t_k are the three neighboring triangles to t. The values λ_i, λ_j and λ_k are the barycentric coordinates of the centroid of t with respect to the centroids of t_i, t_j and t_k. In other words, the numerator is exactly zero when the flow vector associated with the triangle t can be linearly interpolated from the values associated with the neighboring triangles. This discrepancy is then normalized by $|\Delta_{ijk}|$, the area of the triangle formed by connecting the centroids of t_i, t_j and t_k, making the cost akin to a finite-difference approximation of the Laplacian. Similar to the first-order smoothness term, the function $\Psi_\alpha(\cdot)$ is a robust cost function and each term is multiplied by the areas of the three neighboring triangles to impart a non-local character to the cost.

5 Occlusion Reasoning

Since we model an image as a set of triangular pieces that can move independently, we can directly reason about occlusions. A depiction of this process is shown in Figure 2. At each iteration of our algorithm, for each quadrature point in each triangle of I_1, we compute where it appears in the other image I_2. We then determine whether any other triangles overlap it in I_2. For each of these overlapping triangles, we determine whether that triangle offers a better explanation for that location as measured by the data

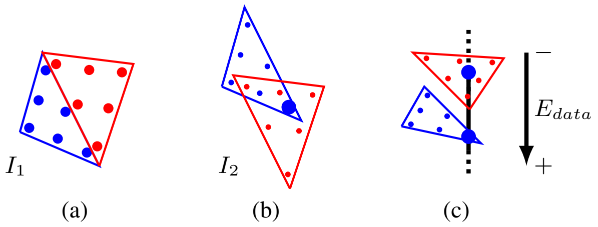

Fig. 2. Depiction of our occlusion term. **(a)** Two triangles and their quadrature points in the tessellation of I_1. **(b)** The triangles are moved to their estimated locations in I_2, where they now overlap. Each quadrature point is processed separately and we have highlighted one quadrature point as an example. **(c)** The data cost is compared for all overlapping triangles at the quadrature point. The quadrature point here has a lower data cost at the same location in the red triangle, and so we mark the quadrature point as occluded.

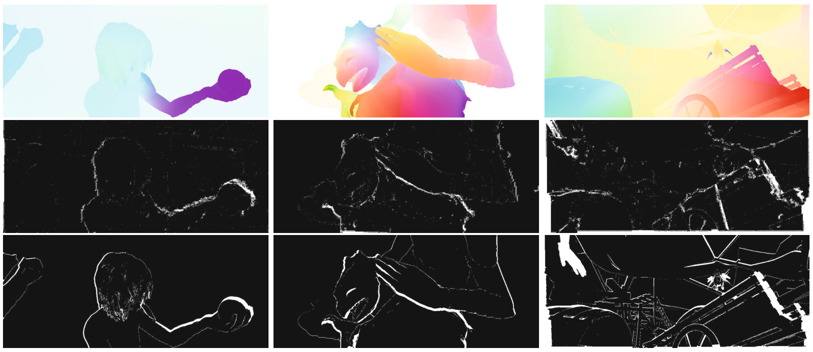

Fig. 3. Examples of our occlusion estimation on MPI-Sintel. During optimization, the occlusion status of each quadrature point in each triangle is directly estimated. For visualization, we label each triangle a value in $[0, 1]$ as the proportion of its quadrature points that are labeled occluded, and then each pixel is labeled based on the triangles that it overlaps. **Top:** Groundtruth flow. **Middle:** Estimated occlusions. **Bottom:** Groundtruth occlusions.

cost (Equation (3)). If a better solution exists, then the quadrature point in question is labeled as occluded. The occluded quadrature points are not included in the evaluation of the data cost. In this way, the data cost function only includes points which are estimated to be unoccluded. Note that these occlusion estimates are generated *directly* from the geometry and from the data cost term; no additional regularization parameters are needed to avoid the trivial solution of labeling all points occluded. An example of our occlusion estimation is show in Figure 3.

Occlusions can be calculated efficiently by rasterizing the triangles to find which pixels they overlap in I_2. When evaluating the occlusion term for a quadrature point, only triangles rasterized to the same pixel need to be considered as potential occluders.

6 Optimization

As is standard, local optimization is carried out within a coarse-to-fine image pyramid [30]. We begin with a zero-valued flow at the coarsest level and iteratively perform local optimization until a local minimum is reached. During this process, image values and gradients are calculated using bicubic interpolation. The resulting solution is then propagated to the next pyramid level where it is used as an initialization and the local optimization is repeated. At each level, a new triangulation is calculated as described in Section 3.

Rather than linearizing the Euler-Lagrange equations [30], we use Newton's method, a second-order optimization algorithm. Newton's method provides flexibility to our framework since any suitably-differentiable function can be substituted for our cost function without changing the optimization scheme. To find the Newton step at each iteration, a sparse linear system must be solved. This is commonly done with an iterative method, such as Successive Over-Relaxation (SOR). Instead, we decompose the Hessian matrix into its Cholesky factorization, after which the linear system can be solved directly. Cholesky factorization is often avoided since it has the potential to use a significant amount of memory, but we have found that the use of a triangulation makes it possible to reduce these memory requirements. First, there are often fewer triangles than pixels, resulting in a smaller linear system. Also, the memory requirement for Cholesky factorization is dependent on the adjacency structure of the matrix, which gives triangulations an advantage since each triangle has only three neighbors rather than four or eight. We have found that the resulting Hessian matrices can be efficiently reordered and factorized using algorithms such as [31].

7 Multi-frame Fusion of Inertial Estimates

A significant challenge for modern optical flow algorithms is when objects move large distances. This is especially true when objects move either into or out of frame, for which there are no matches. In this case, it is often not possible to estimate the motion of these pixels from two-frame optical flow. In this section, we address this by proposing a simple method of incorporating temporal information from adjacent frames.

7.1 Inertial Estimates

Let $[t \rightarrow (t+1)]$ denote the estimated flow between frames t and $t+1$, and suppose that we also have access to frames $t-1$ and $t+2$. If it is assumed that objects move at a constant velocity (i.e., they are carried by inertia) and move parallel to the image plane, then an estimate of the motion from $[t \rightarrow (t+1)]$ is given by $-[t \rightarrow (t-1)]$, which is found by computing the flow from t to $t-1$ and negating it, as shown in Figure 4. Similarly, another estimate can be found using frame $t+2$ as $\frac{1}{2}[t \rightarrow (t+2)]$. We call these "inertial estimates" since they provide an estimate of the flow by assuming that inertia moves all objects at a constant velocity. All three inertial estimates are computed independently, using the same optical flow algorithm.

Of course, these estimates will, on average, be inferior to using $[t \rightarrow (t+1)]$ directly. However, if an object is visible in frame t and moves out of frame in $t+1$, then it may

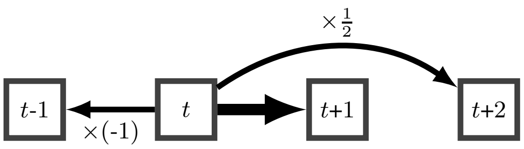

Fig. 4. Inertial flow estimates used in multi-frame fusion. In addition to using the two-frame estimate $[t \to (t+1)]$ directly, we also estimate the flow from $[t \to (t-1)]$ and from $[t \to (t+2)]$. These two estimates are then multiplied by the factors -1 and $\frac{1}{2}$, respectively, to give an estimate of the desired flow $[t \to (t+1)]$. All three flow estimates are then fused using a classifier.

still be visible in $t-1$ and so $-[t \to (t-1)]$ will likely give a better estimate for that part of the image. Similarly, using the estimate $\frac{1}{2}[t \to (t+2)]$ will provide an additional source of information.

7.2 Classifier-Based Fusion

The three inertial estimates $[t \to (t+1)]$, $\frac{1}{2}[t \to (t+2)]$ and $-[t \to (t-1)]$ must be fused. We do so by training a random forest classifier whose output tells us which estimate for each pixel is predicted to have the lowest error. We use the following features:

- The tail probability for the Cauchy distribution used in the match cost $\mathcal{D}(\cdot)$. This value varies from 0 to 1 with larger values indicating a better match. The index of the flow estimate with the best score, and its associated score, are also used.
- Each pixel in frame t is projected forward via the flow estimate and then projected back using the backward flow. The Euclidean norm of this discrepancy vector is used as a feature for each flow estimate. The index of the flow estimate with the smallest discrepancy and its corresponding value are also included as features.
- The flow estimates u and v, and the magnitudes $\sqrt{u^2 + v^2}$.
- The multiplicative offset $m(\mathbf{x})$ at each pixel.
- For every pixel, an indicator of whether the pixel is estimated to be occluded.
- The (x, y) location of each pixel.

This results in a total of 27 features. For each dataset that we evaluated our methods on, we sampled a number of points uniformly at random from the associated training images such that the resulting dataset had $\sim 10^6$ observations.

In this classification problem, not all data points should be counted equally. In particular, a misclassification is more costly when the three inertial estimates have very different errors. To take this into account, each data point was weighted by the difference between the lowest endpoint error of all three flow estimates and the mean of the other two. This weighting indicates how important each datapoint is. We then trained a random forest classifier with 500 trees using MATLAB's TreeBagger class. An example of our fusion is shown in Figure 5 on the MPI-Sintel dataset [5]. As a final step in our procedure, a median filter of size 15×15 was applied.

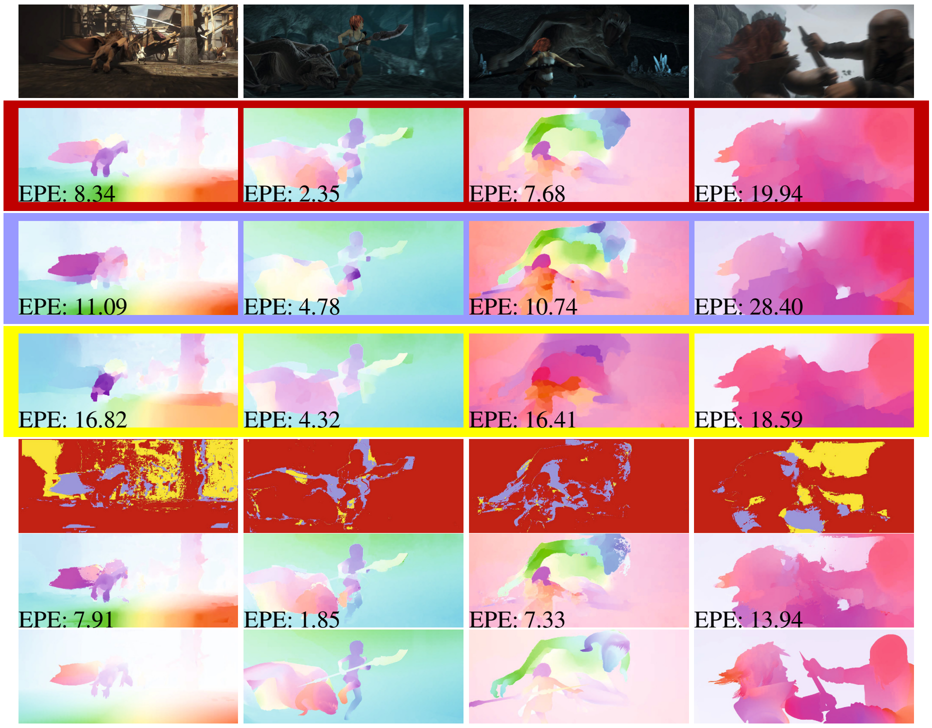

Fig. 5. Examples of our multi-frame fusion from the MPI-Sintel Final training set. **Top row:** Frame at time t. **Rows 2-4:** Inertial estimates of the flow $[t \rightarrow t+1]$, $-[t \rightarrow t-1]$, and $\frac{1}{2}[t \rightarrow t+2]$. **Row 5:** Fusion classification for each pixel. Color indicates the estimate used at each pixel. Colors correspond to the border colors of the inertial estimates. **Row 6:** Fused flow estimate. **Bottom row:** Groundtruth flow. For all flow estimates, the endpoint error is printed in the image.

8 Experiments

We evaluate our algorithm on three datasets. The free parameters of our method are $\tau_0, \tau_1, \tau_2, \tau_3$, the value of α used in the smoothness terms $\mathcal{S}_1(\cdot)$ and $\mathcal{S}_2(\cdot)$, and the spacing of the uniform grid used in the triangulation. For $\mathcal{S}_3(\cdot)$, we set $\alpha = 0.5$ for all experiments. Parameters were chosen for each dataset using a small-scale grid search on the training data.

We use a scale of 0.95 between pyramid levels – resulting in around 60 levels – and used 10 Cholesky-based iterations of Newton's method at each level.

Our basic method is denoted as TF (TriFlow), and when occlusion estimation, multi-frame fusion and median filtering are used, they are denoted as “O” (occlusion), “F” (fusion) and “M” (median filtering), respectively. Our final method with all components is thereby denoted as TF+OFM.

8.1 Middlebury

We begin with the Middlebury dataset [4] since it is a standard benchmark for optical flow, although it has only small and simple motions. For this dataset, the parameters were set to $\tau_0 = 0, \tau_1 = 3.5, \tau_2 = 0, \tau_3 = 25, \alpha = 0.36$, and a small grid spacing of 2 pixels was used in order to capture the small details in this dataset.

Results on the test dataset are given in Table 1. Note that we did not evaluate our multi-frame fusion since the dataset was too small for a reliable classifier to be trained. Our results are comparable to other similar coarse-to-fine methods such as DeepFlow [27]. Our occlusion estimation provides little benefit in this case, since the dataset has very small occlusion regions.

Table 1. Endpoint error on the Middlebury test dataset. Our results are comparable with similar coarse-to-fine methods

	Army	Mequon	Scheff.	Wooden	Grove	Urban	Yosemite	Teddy	mean
TF+OM	0.10	0.22	0.36	0.20	0.98	0.56	0.16	0.76	0.42
Layers++ [10]	0.08	0.19	0.20	0.13	0.48	0.47	0.15	0.46	0.27
MDP-Flow2 [12]	0.09	0.19	0.24	0.16	0.74	0.46	0.12	0.78	0.35
DeepFlow [27]	0.12	0.28	0.44	0.26	0.81	0.38	0.11	0.93	0.42
LDOF [24]	0.12	0.32	0.43	0.45	1.01	0.10	0.12	0.94	0.56

8.2 MPI-Sintel

The MPI-Sintel dataset [5] is a large, difficult dataset that includes large displacements, significant occlusions and atmospheric effects. Parameters were set to $\tau_0 = 0.5, \tau_1 = 2.0, \tau_2 = 0, \tau_3 = 100, \alpha = 0.6$, and the grid spacing was set to 5 pixels.

Results on the MPI-Sintel test dataset are given in Table 2. As of this writing, our method is ranked 2nd among all submissions on the Final dataset and it outperforms all other published methods in terms of endpoint error. The results are especially good for unmatched pixels which are helped by our occlusion term and multi-frame fusion. In particular, on the Final dataset the occlusion term improves the error on unmatched pixels by 6.4% and the fusion improves it by an additional 7.2%.

Several examples of results from our multi-frame fusion for the Final version of the training dataset are shown in Figure 5. As we would expect, the inertial estimates that the classifier selects are spatially localized around the edges of objects where occlusions occur. In all cases, the multi-frame fusion significantly reduces the endpoint error.

Figure 3 shows several examples of the occlusion estimates on images from MPI-Sintel. During optimization, the occlusion status of each quadrature point in each triangle is estimated. For visualization, we label each triangle a value in $[0, 1]$ as the proportion of its quadrature points that are labeled as occluded, and then each pixel is labeled based on the triangles that it overlaps. The occlusion term is able to estimate the occlusions accurately, which results in reduced error.

Table 2. Results on the MPI-Sintel test set. Our algorithm outperforms all other published results in terms of endpoint error (EPE) on the Final dataset. The largest change is on unmatched pixels due to our occlusion estimation and multi-frame fusion.

	Final			Clean		
	EPE	matched	unmatched	EPE	matched	unmatched
TF+OFM	**6.727**	3.388	**33.929**	4.917	1.874	29.735
TF+OF	6.780	3.436	34.029	4.986	1.937	29.857
TF+O	7.164	3.547	36.657	5.357	2.033	32.474
TF	7.493	3.609	39.170	5.723	2.077	35.471
DeepFlow [27]	7.212	**3.336**	38.781	5.377	1.771	34.751
AggregFlow [32]	7.329	3.696	36.929	**4.754**	**1.694**	**29.685**
FC-2Layers-FF [16]	8.137	4.261	39.723	6.781	3.053	37.144
MDP-Flow2 [12]	8.445	4.150	43.430	5.837	1.869	38.158
LDOF [24]	9.116	5.037	42.344	7.563	3.432	41.170

8.3 KITTI

The KITTI dataset [6] consists of grayscale images taken from a moving vehicle. We used the parameter settings $\tau_0 = 0.05, \tau_1 = 0.02, \tau_2 = 7, \tau_3 = 125, \alpha = 0.6$, and the grid spacing was set to 5 pixels.

On the KITTI test dataset, error is measured as the percentage of pixels with an endpoint error greater than 3, in addition to the standard endpoint error. Our results on this dataset are given in Table 3. This dataset is quite different than MPI-Sintel: the images are grayscale and have low contrast and the motions are often dominated by that of the camera. Top-performing methods on this dataset take advantage of these properties by using better features such as census transforms and more information such as stereo and epipolar information [33]. However, our results are comparable to similar coarse-to-fine approaches such as DeepFlow [27], especially for endpoint error (which the fusion classifier was trained to minimize).

We also evaluate the effect of our occlusion and fusion terms on a validation set from the training images. For this, 100 training images were used to train a fusion classifier and evaluation was done on remaining 94 images. Results are show in Table 4. Both the occlusion and multi-frame fusion terms significantly improve results, as measured by either endpoint error or the percentage of pixels with and endpoint error more than 3.

8.4 Timing

Timing was evaluated on a laptop with a 1.80 GHz Intel Core i5 processor and 4 GB of RAM. The typical time taken for two-frame flow estimation on a 1024×436 image from MPI-Sintel (including all setup and feature matching, but excluding multi-frame fusion) was 500 seconds. About half of this time is spent evaluating the cost function within Newton's method, and another 20% is spent solving linear systems. Much of our approach can be sped up through parallelization. For example, the cost function evaluation, running the algorithm on all three inertial estimates, and the random forest fusion are all trivially-parallelizable.

Table 3. Results on the KITTI test set. We show both the endpoint error (EPE) and the percentage of pixels with an EPE more than 3, for all pixels as well as non-occluded pixels.

	EPE (all)	EPE (not occ.)	$\% > 3$ (all)	$\% > 3$ (not occ.)
TF+OFM	5.0	2.0	18.46%	10.22%
PCBP-Flow [33]	2.2	0.9	8.28%	3.64%
DeepFlow [27]	5.8	1.5	17.79%	7.22%
LDOF [24]	12.4	5.6	31.39%	21.93%
DB-TV-L1 [34]	14.6	7.9	39.25%	30.87 %

Table 4. Results on a validation set from the KITTI training dataset. The occlusion estimation term and multi-frame fusion significantly improve results.

	EPE	$\% > 3$
TF+OFM	4.23	16.43%
TF+OF	4.32	16.62%
TF+O	5.29	16.91%
TF	6.89	19.96%

9 Conclusion

This paper presents a novel framework for estimating optical flow based on a triangulation of the image which improves results in difficult regions due to occlusions and large motions. We use a geometric model that allows us to directly account for occlusion effects. We also present a method that exploits temporal information from adjacent frames by acquiring several flow estimates and fusing them via a classifier. Together, these contributions result in state-of-the-art performance on the MPI-Sintel dataset. Our approach was evaluated on a range of datasets and the results demonstrate that the proposed enhancements have a significant impact on the quality of the resulting flow.

References

1. Horn, B.K., Schunck, B.G.: Determining optical flow. Artificial Intelligence 17(1), 185–203 (1981)
2. Black, M.J., Anandan, P.: The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. CVIU 63(1), 75–104 (1996)
3. Sun, D., Roth, S., Black, M.J.: Secrets of optical flow estimation and their principles. In: CVPR (2010)
4. Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. IJCV 92(1), 1–31 (2011)
5. Butler, D.J., Wulff, J., Stanley, G.B., Black, M.J.: A naturalistic open source movie for optical flow evaluation. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part VI. LNCS, vol. 7577, pp. 611–625. Springer, Heidelberg (2012)
6. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: CVPR (2012)
7. Lempitsky, V., Roth, S., Rother, C.: Fusionflow: Discrete-continuous optimization for optical flow estimation. In: CVPR, pp. 1–8. IEEE (2008)
8. Xu, L., Chen, J., Jia, J.: A segmentation based variational model for accurate optical flow estimation. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 671–684. Springer, Heidelberg (2008)
9. Strecha, C., Fransens, R., Van Gool, L.: A probabilistic approach to large displacement optical flow and occlusion detection. In: Comaniciu, D., Mester, R., Kanatani, K., Suter, D. (eds.) SMVP 2004. LNCS, vol. 3247, pp. 71–82. Springer, Heidelberg (2004)
10. Sun, D., Sudderth, E.B., Black, M.J.: Layered image motion with explicit occlusions, temporal consistency, and depth ordering. In: NIPS, pp. 2226–2234 (2010)
11. Glocker, B., Heibel, T.H., Navab, N., Kohli, P., Rother, C.: TriangleFlow: Optical flow with triangulation-based higher-order likelihoods. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part III. LNCS, vol. 6313, pp. 272–285. Springer, Heidelberg (2010)

12. Xu, L., Jia, J., Matsushita, Y.: Motion detail preserving optical flow estimation. PAMI 34(9), 1744–1757 (2012)
13. Kim, T.H., Lee, H.S., Lee, K.M.: Optical flow via locally adaptive fusion of complementary data costs. In: ICCV (2013)
14. Sun, D., Liu, C., Pfister, H.: Local layering for joint motion estimation and occlusion detection. In: CVPR (2014)
15. Volz, S., Bruhn, A., Valgaerts, L., Zimmer, H.: Modeling temporal coherence for optical flow. In: ICCV, pp. 1116–1123. IEEE (2011)
16. Sun, D., Wulff, J., Sudderth, E.B., Pfister, H., Black, M.J.: A fully-connected layered model of foreground and background flow. In: CVPR (2013)
17. Mac Aodha, O., Humayun, A., Pollefeys, M., Brostow, G.J.: Learning a confidence measure for optical flow. IEEE Transactions on Pattern Analysis and Machine Intelligence 35(5), 1107–1120 (2013)
18. Jung, H.Y., Lee, K.M., Lee, S.U.: Toward global minimum through combined local minima. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part IV. LNCS, vol. 5305, pp. 298–311. Springer, Heidelberg (2008)
19. Chang, H.S., Wang, Y.C.F.: Superpixel-based large displacement optical flow. In: ICIP, pp. 3835–3839 (2013)
20. Negahdaripour, S.: Revised definition of optical flow: Integration of radiometric and geometric cues for dynamic scene analysis. PAMI 20(9), 961–979 (1998)
21. Donoser, M., Schmalstieg, D.: Discrete-continuous gradient orientation estimation for faster image segmentation. In: CVPR (2014)
22. Cowper, G.: Gaussian quadrature formulas for triangles. International Journal for Numerical Methods in Engineering 7(3), 405–408 (1973)
23. Sun, D., Roth, S., Lewis, J.P., Black, M.J.: Learning optical flow. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 83–97. Springer, Heidelberg (2008)
24. Brox, T., Malik, J.: Large displacement optical flow: descriptor matching in variational motion estimation. PAMI 33(3), 500–513 (2011)
25. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR, vol. 1, pp. 886–893. IEEE (2005)
26. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. In: VISAPP, pp. 331–340 (2009)
27. Weinzaepfel, P., Revaud, J., Harchaoui, Z., Schmid, C.: Deepflow: Large displacement optical flow with deep matching. In: ICCV (2013)
28. Byrne, J., Shi, J.: Nested shape descriptors. In: ICCV, pp. 1201–1208. IEEE (2013)
29. Werlberger, M., Pock, T., Bischof, H.: Motion estimation with non-local total variation regularization. In: CVPR, pp. 2464–2471. IEEE (2010)
30. Brox, T., Bruhn, A., Papenberg, N., Weickert, J.: High accuracy optical flow estimation based on a theory for warping. In: Pajdla, T., Matas, J(G.) (eds.) ECCV 2004. LNCS, vol. 3024, pp. 25–36. Springer, Heidelberg (2004)
31. Amestoy, P.R., Davis, T.A., Duff, I.S.: Algorithm 837: Amd, an approximate minimum degree ordering algorithm. ACM Trans. Math. Softw. 30(3), 381–388 (2004)
32. Fortun, D., Bouthemy, P., Kervrann, C.: Aggregation of local parametric candidates with exemplar-based occlusion handling for optical flow. arXiv preprint arXiv:1407.5759v1
33. Yamaguchi, K., McAllester, D., Urtasun, R.: Robust monocular epipolar flow estimation. In: CVPR, pp. 1862–1869. IEEE (2013)
34. Zach, C., Pock, T., Bischof, H.: A duality based approach for realtime TV-L^1 optical flow. In: Hamprecht, F.A., Schnörr, C., Jähne, B. (eds.) DAGM 2007. LNCS, vol. 4713, pp. 214–223. Springer, Heidelberg (2007)