Locally Oriented Optical Flow Computation

Yan Niu, Anthony Dick, and Michael Brooks

***Abstract*—This paper proposes the use of an adaptive locally oriented coordinate frame when calculating an optical flow field. The coordinate frame is aligned with the least curvature direction in a local window about each pixel. This has advantages to both fitting the flow field to the image data and in imposing smoothness constraints between neighboring pixels. In terms of fitting, robustness is obtained to a wider variety of image motions due to the extra invariance provided by the coordinate frame. Smoothness constraints are naturally propagated along image boundaries which often correspond to motion boundaries. In addition, moving objects can be efficiently segmented in the least curvature direction. We show experimentally the benefits of the method and demonstrate robustness to fast rotational motion, such as what often occurs in human motion.**

***Index Terms*—Directional derivative, image structure, intrinsic direction detection, motion estimation, optical flow.**

I. Introduction

OPTICAL flow estimation, which is the estimation of apparent motion between images at each pixel, is a classic ill-posed inverse problem in computer vision. It is fundamental to many techniques in motion estimation, tracking, segmentation, and compression.

Most approaches recover optical flow as a vector field that best satisfies a set of constraints on data fitting and smoothness. Data constraints are used to infer each pixel's flow vector from the image data, assuming that a certain intensity signature of the pixel (e.g., brightness [1], gradient [2], or curvature [3]) remains invariant or varies predictably over time. Smoothness constraints regularize each pixel's flow by its neighbors' flow, assuming that the flow field is smooth to some extent. These constraints can be further classified as local or nonlocal, depending on whether the regularization is applied only to the immediate neighbors of a pixel or a larger surrounding window [4].

Except for a few examples that use a log-polar coordinate system (e.g., in [5]–[7]), most flow computation constraints are modeled in the horizontal–vertical coordinate frame formed by the image grid. This coordinate system is convenient for computation but often fails to represent the intrinsic image structure, which is due to the projection of a 3-D scene onto an image plane. In other words, the directions and magnitude values of local intensity edges and gradients in natural images are seldom aligned with the image grid. Intensity variation information in particular provides important prior knowledge for motion boundary prediction as the motion boundary often coincides with the intensity edge.

To overcome such inflexibility, this paper proposes using the coordinate frame determined by the underlying image structure instead of the uniform image grid to model the flow computation constraints. Specifically, we associate each pixel with the local reference frame spanned by the unit vectors in its edge and normal directions. We refer to this adaptive coordinate system as the intrinsic or structure-oriented (SO) coordinate[1] and the traditional coordinate system as the grid-based coordinate. We shall show that the SO coordinate benefits motion boundary preservation, rotation robustness, and motion outlier exclusion.

In the presence of a motion boundary, the local motion field has the least significant variation in the local edge direction, and the most significant variation in the normal direction, which should be respected in the smoothness constraints to preserve motion boundaries. For this reason, flow regularization should be directed to the intrinsic directions. This can be naturally achieved by modeling the local smoothness constraints, which generally constrain flow variation in axial directions, in the SO coordinate system. Many previous works (e.g., in [8]–[10]) aim at the same goal but obtain the flow directional variation by a different approach: They project the flow gradient onto the detected intrinsic directions. This kind of projection implicitly assumes that the motion field is continuously differentiable, and the local regularization constraint is in fact imposed to the horizontal–vertical variation of the flow. If the continuous differentiability assumption is valid, the projection and the SO coordinate frame have a similar effect. Otherwise, as we shall show in this paper, the projection is not guaranteed to obtain the directional variation accurately. This sensitivity to motion differentiability is bypassed in the structure-based coordinate frame.

Another significant difference between this paper and the above previous works is that this paper embeds the intrinsic directions not merely into the local smoothness constraint but also into the data constraints and the nonlocal smoothness constraint, through the SO coordinate frame.

Manuscript received January 10, 2011; revised June 30, 2011, October 18, 2011 and November 02, 2011; accepted November 02, 2011. Date of publication December 01, 2011; date of current version March 21, 2012. This work was supported in part by the National Natural Science Foundation of China under Grant NSFC61103090 and the Australian Research Council under Grant DP1094764. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Brendt Wohlberg.

Y. Niu is with the State Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education, College of Computer Science and Technology, JiLin University, Changchun 130012, China (e-mail: niuyan@jlu.edu.cn).

A. Dick and M. Brook are with School of Computer Science, the University of Adelaide, Adelaide, SA 5005, Australia.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2011.2177847

[1]A similar concept is the gauge coordinate, which has the axes aligned with the image gradient and the perpendicular direction. In practice, the gradient is approximated by finite differencing in the x- and y-directions. As we shall show later, if the intensity function is not continuously differentiable, the approximated gradient does not necessarily coincide with the true edge normal. In this paper, the axial directions of the intrinsic coordinate frame are detected by applying the original definition of directional variation. Therefore, strictly speaking, it is not the gauge coordinate.

Because one axis of the local reference frame is aligned with the edge, two individually moving objects are naturally segmented to each side of the edge axis. Therefore, orthogonal to the edge axis, one object is located in the positive side and the other object in the negative side. We present a simple scheme to decide which side belongs to the motion inlier object. Based on this, motion outliers can be excluded by simply checking the sign of the pixel coordinates. In previous related works (e.g., in [11]–[14]), outliers are suppressed in the nonlocal smoothness constraints by computing the pairwise weights over the whole "nonlocal" window. This either incurs massive storage or intensive computation load.

In addition, the grid-based reference frame is fixed over time, whereas the SO frame changes with the structure. Therefore, if the object undergoes fast rotation, the image gradient expressed in the grid directions changes between frames, but its counterpart expressed in the intrinsic directions is invariant. The rotational invariance of this descriptor thus leads to data constraints that are robust to rotation, which is desirable in real applications such as human motion estimation.

The idea of a SO coordinate frame is independent of the choice of data and smoothness constraints and can therefore be incorporated into many existing flow algorithms that are formulated in this way. The formulation and optimization described in this paper can hence be applied, with some changes in detail, to a range of previously proposed cost functions, where examples of which are given in Section II.

The rest of this paper explores the construction of the SO coordinate frame, the formulation of an objective functional based on it, and the numerical minimization process. Section II takes several fundamental flow constraints as examples to theoretically analyze the desirable features of SO coordinates. In Section III, we present a new approach to computing the directional derivatives and estimating the local edge direction, based on which we construct the intrinsic reference frame. Section IV derives the numerical solution for the example formulation. Four sets of experiments are conducted to quantitatively and qualitatively evaluate the approaches proposed in this paper, and the results are analyzed in Section V. The contribution of this paper is summarized by Section VI.

II. Benefits of the SO Coordinate Frame

Given the brightness function E of an image sequence at the horizontal, vertical, and temporal positions (x, y, t), we aim to infer the perceived motion (i.e., optical flow) (u, v) from the spatiotemporal variation of $E(x, y, t)$ as an estimation of the physical motion $((dx/dt), (dy/dt))$. For the moment, we assume that the unit vectors of the local edge and normal directions are known (the detection of them will be addressed in Section III) and denote them by $\boldsymbol{d}$ and $\boldsymbol{n}$, respectively. These two vectors construct a local reference frame at the current pixel. We index the axes of this coordinate frame by ζ and η. For simplicity, the SO coordinate frame is referred to as the $\zeta\eta$-frame, and the traditional grid-based coordinate frame is referred to as the xy-frame. The $\zeta\eta$-frame degenerates to the xy-frame if $\boldsymbol{d}$ and $\boldsymbol{n}$ are in the horizontal and vertical directions at each pixel. To respect oblique motion, we assume that the flow field is locally affine

$$\begin{aligned}\tilde{u} &= u + \alpha_1\zeta + \alpha_2\eta \\ \tilde{v} &= v + \alpha_3\zeta + \alpha_4\eta\end{aligned} \tag{1}$$

where (u, v) is the flow vector of the current pixel, $(\tilde{u}, \tilde{v})$ is the flow vector of a pixel in the local area, and deformation parameters α_i's are locally constant. Note that $\alpha_1 = (\partial u/\partial \boldsymbol{d})$, $\alpha_2 = (\partial u/\partial \boldsymbol{n})$, $\alpha_3 = (\partial v/\partial \boldsymbol{d})$, and $\alpha_4 = (\partial v/\partial \boldsymbol{n})$.

In this section, we illustrate how the $\zeta\eta$-frame enhances the motion robustness, discontinuity-preserving ability, and efficiency of the flow recovery. Although the illustration uses a number of basic optical flow constraints as examples, the discussion can be generalized to more sophisticated constraints that are based on these. As the example constraints are expressed by mathematical equations whose left-hand sides are "deviations" and right-hand sides are zeros, we define the associated energy terms by applying a penalizing function to the left-hand sides of these equations. The combination of these energy terms forms the example cost function to be minimized in Section IV.

A. Brightness Constancy Constraint

The most commonly used data constraint is the brightness constancy assumption (BCA), which states that $E(x+u, y+v, t+1)$ remains the same value as $E(x, y, t)$ or $\dot{E} = 0$.[2] Linearized by the first-order Taylor expansion, this data constraint is conventionally modeled by the following mathematical equation:

$$\dot{E} \approx E_x u + E_y v + E_t = 0. \tag{2}$$

Here and throughout this paper, the subscripts of x, y, $\boldsymbol{d}$, $\boldsymbol{n}$, and t denote the corresponding partial derivatives.

The BCA in the $\zeta\eta$-coordinates remains the same as in the xy-coordinates. That is, the flow vector (u, v) is constrained to minimize the change of the brightness. The associated data energy term is defined by

$$\mathfrak{E}_{\dot{E}} = \psi(\dot{E}) \tag{3}$$

where $\psi()$ represents the penalizing function. The penalizers $\psi(\chi) = \chi^2$ (e.g., in [15]), $\psi(\chi) = |\chi|$ (e.g., in [16]) or the Charbonnier $\psi(\chi) = \sqrt{\chi^2 + \epsilon}$ (e.g., in [17]) are common choices. Other robust norm functions such as the Lorentzian [18] or the Huber norm [19], [20] also have demonstrated effectiveness.

B. Gradient Constancy Constraint

In order to improve the robustness of the flow recovery to lighting changes, many works adopt an additional data constraint based on the gradient constancy assumption (GCA). For example, Zimmer *et al.* [17] use

$$\begin{aligned}\frac{dE_x}{dt} &\approx E_{xx}u + E_{xy}v + E_{xt} = 0 \\ \frac{dE_y}{dt} &\approx E_{xy}u + E_{yy}v + E_{yt} = 0.\end{aligned} \tag{4}$$

[2]In this paper, we use $(\dot{\chi})$ as a compact notation for the total temporal derivative $(d\chi/dt)$.

Unfortunately, the GCA is invalid in the presence of rotation [21]. To remedy this disadvantage, Weickert *et al.* [22] suggest using the gradient magnitude, which is rotation invariant. However, as the gradient magnitude discards one dimension of information, it is less effective if rotation is not involved. The SIFT feature [23], which is rotation and scale invariant, has been integrated into the flow computation in [24] for scene matching. Yet, the SIFT feature is not differentiable and hence unsuitable for variational flow computation. The invariance of higher order derivatives, such as Laplacian and the determinant of Hessian, has been discussed in [22]. Clearly, such features are more sensitive to noise compared with the gradient.

Rotational robustness can be straightforwardly achieved in the $\zeta\eta$-frame, where the image gradient is $\nabla_d E = [E_{\boldsymbol{d}}, E_{\boldsymbol{n}}]^T$, and the GCA reads $(d\nabla_d E/dt) = 0$. The validity of this assumption is robust to rotation. Moreover, $\nabla_d E$ also has the desirable probabilities of differentiability and noise robustness. Its total temporal derivative can be linearized similarly to [17], i.e.,

$$\begin{aligned}\frac{dE_{\boldsymbol{d}}}{dt} &\approx E_{x\boldsymbol{d}}u + E_{y\boldsymbol{d}}v + E_{t\boldsymbol{d}}\\ \frac{dE_{\boldsymbol{n}}}{dt} &\approx E_{x\boldsymbol{n}}u + E_{y\boldsymbol{n}}v + E_{t\boldsymbol{n}}.\end{aligned} \tag{5}$$

Our empirical study also shows that the GCA in the $\zeta\eta$-frame quantitatively outperforms the GCA in the xy-frame. However, (5) does not take oblique motion into consideration. We switch the order of taking the spatioemporal derivatives in (5), which leads to

$$\begin{aligned}\dot{E}_{\boldsymbol{d}} &= \frac{\partial}{\partial \boldsymbol{d}}\left(\frac{dE}{dt}\right) = 0\\ \dot{E}_{\boldsymbol{n}} &= \frac{\partial}{\partial \boldsymbol{n}}\left(\frac{dE}{dt}\right) = 0\end{aligned} \tag{6}$$

and linearize $\dot{E}_{\boldsymbol{d}}$ and $\dot{E}_{\boldsymbol{n}}$ by the chain rule into

$$\begin{aligned}\dot{E}_{\vec{\boldsymbol{d}}} &\approx \frac{\partial(E_x u + E_y v + E_t)}{\partial \vec{\boldsymbol{d}}}\\ &\approx E_{x\vec{\boldsymbol{d}}}u + E_x u_{\vec{\boldsymbol{d}}} + E_{y\vec{\boldsymbol{d}}}v + E_y v_{\vec{\boldsymbol{d}}} + E_{t\vec{\boldsymbol{d}}}\\ \dot{E}_{\vec{\boldsymbol{n}}} &\approx \frac{\partial(E_x u + E_y v + E_t)}{\partial \vec{\boldsymbol{n}}}\\ &\approx E_{x\vec{\boldsymbol{n}}}u + E_x u_{\vec{\boldsymbol{n}}} + E_{y\vec{\boldsymbol{n}}}v + E_y v_{\vec{\boldsymbol{n}}} + E_{t\vec{\boldsymbol{n}}}.\end{aligned} \tag{7}$$

Equation (7) extends the GCA to oblique motion: If the deformation parameters $u_{\boldsymbol{d}}$, $u_{\boldsymbol{n}}$, $v_{\boldsymbol{d}}$, and $v_{\boldsymbol{n}}$ of the oblique motion (u, v) are zeros, (7) degenerates to (5). The associated data energy terms are defined by

$$\mathfrak{E}_{\dot{E}_{\boldsymbol{d}}} = \psi(\dot{E}_{\boldsymbol{d}}), \quad \mathfrak{E}_{\dot{E}_{\boldsymbol{n}}} = \psi(\dot{E}_{\boldsymbol{n}}). \tag{8}$$

C. Local Smoothness Constraint

The smoothness constraint for affine flow has been discussed by Ju *et al.* [25], Nir *et al.* [26], and by Trobin *et al.* [27]. We compare in the two coordinate systems the behavior of the smoothness term by Nir *et al.* as it is closest to the general smoothness terms often used in flow calculation. In the xy-frame, letting $\boldsymbol{A} = [u, v, u_x, v_x, u_y, v_y]^T$ collect the affine motion parameters, the local smoothness constraint by Nir *et al.* assumes that the spatiotemporal variation of $\boldsymbol{A}$ is zero [26] as follows:

$$\sum_{i=1}^{6} \|\tilde{\nabla}\boldsymbol{A}_i\|^2 = 0 \tag{9}$$

where $\tilde{\nabla}$ means the spatiotemporal gradient and $\boldsymbol{A}_i$ means the ith element of $\boldsymbol{A}$. In the $\zeta\eta$-frame, let vector $\boldsymbol{m} = [u, v, u_{\boldsymbol{d}}, v_{\boldsymbol{d}}, u_{\boldsymbol{n}}, v_{\boldsymbol{n}}]^T$ collect the motion parameters; thus, $\nabla_d \boldsymbol{m}$ measures the spatial variation of $\boldsymbol{m}$. The smoothness constraint becomes

$$\sum_{i=1}^{6} \|\nabla_d \boldsymbol{m}_i\|^2 = 0. \tag{10}$$

Constraint (10) is more suitable than constraint (9) to preserve motion boundaries, in that $\boldsymbol{d}$ (resp. $\boldsymbol{n}$) is the most (resp. least) smooth direction; hence, the flow smoothness should be enforced or inhibited in these directions rather than in the x- and y-directions. To steer the smoothness differently in the edge direction and edge normal, we employ the following local smoothness constraints in our simulation:

$$\|\boldsymbol{m}_{\boldsymbol{d}}\|_2^2 = 0, \qquad \|\boldsymbol{m}_{\boldsymbol{n}}\|_2^2 = 0. \tag{11}$$

The associated smoothness energy term is defined by

$$\mathfrak{E}_s = \psi\left(\|\boldsymbol{m}_{\boldsymbol{d}}\|_2\right) + \psi\left(\|\boldsymbol{m}_{\boldsymbol{n}}\|_2\right). \tag{12}$$

The local intrinsic directions have been exploited for motion boundary preservation in many previous works. For example, Nagel and Enkelmann [8] estimate $\boldsymbol{n}$ by the gradient vector $\nabla E = [E_x, E_y]^T$ and $\boldsymbol{d}$ by the perpendicular vector $\nabla E^{\perp}$. Sun *et al.* [9] detect the local intrinsic directions by the eigenvectors of the local structure tensor. As pointed out in [10] by Zimmer *et al.*, both Nagel and Enkelmann [8] and Sun *et al.* [9] have the smoothness terms in the following form[3]:

$$\begin{aligned}&\omega_1(E, u, v)\psi(u_{\vec{\boldsymbol{d}}}) + \omega_2(E, u, v)\psi(u_{\vec{\boldsymbol{n}}})\\ &\qquad + \omega_3(E, u, v)\psi(v_{\vec{\boldsymbol{d}}}) + \omega_4(E, u, v)\psi(v_{\vec{\boldsymbol{n}}})\end{aligned} \tag{13}$$

where $u_{\boldsymbol{d}}$ is obtained by projecting ∇u to the direction $\boldsymbol{d}$ and similarly for $u_{\boldsymbol{n}}$, $v_{\boldsymbol{d}}$, and $v_{\boldsymbol{n}}$. The weight functions ω_i balance the penalization of the flow variation in the directions of $\boldsymbol{d}$ and $\boldsymbol{n}$. Note that the smoothness constraint is ultimately applied to the horizontal–vertical variation of the flow field. Our local smoothness constraint modeled in the $\zeta\eta$-frame is directly applied to the flow variation in the intensity edge and normal directions. In the extreme case that the deformation parameters are zeros, the right-hand side of (12) degenerates to

$$\psi\left(\sqrt{u_{\boldsymbol{d}}^2 + v_{\boldsymbol{d}}^2}\right) + \psi\left(\sqrt{u_{\boldsymbol{n}}^2 + v_{\boldsymbol{n}}^2}\right) \tag{14}$$

which, at first glance, has a similar form to expression (13). One major difference is that we obtain $\boldsymbol{d}$ and $\boldsymbol{n}$ by a directional convolution instead of predicting from the horizontal–vertical finite

[3]Note that Zimmer *et al.* proposes using a different set of intrinsic directions and different penalizing functions in their work [10].

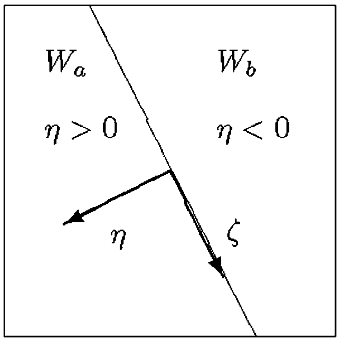

Fig. 1. The line in the edge direction segments the surrounding window W into two subregions W_a and W_b, which are above and below the line, respectively. In the $\zeta\eta$-frame, $W_a = \{(\zeta,\eta)|\eta < 0\}$ and $W_b = \{(\zeta,\eta)|\eta < 0\}$.

differencing of E, and obtain $[u_{\boldsymbol{d}}, u_{\boldsymbol{n}}, v_{\boldsymbol{d}}, v_{\boldsymbol{n}}]$ by the original definition of directional derivatives instead of projecting ∇u and ∇v to $\boldsymbol{d}$ and $\boldsymbol{n}$, which we explain in Section III.

D. Nonlocal Smoothness Constraint

Recently, a third type of constraints, which infers the flow vector from the statistics of the surrounding nonlocal window, has caused great research attention [11]–[14]. The nonlocal regularization terms generally have the following form:

$$\sum_{(\tilde{x},\tilde{y})\in W} \omega_1(\tilde{x},\tilde{y})\psi\left(u(x,y) - u(\tilde{x},\tilde{y})\right) + \sum_{(\tilde{x},\tilde{y})\in W} \omega_2(\tilde{x},\tilde{y})\psi\left(v(x,y) - v(\tilde{x},\tilde{y})\right) \quad (15)$$

where W is the nonlocal window. Previous works have shown that if $\psi(\chi) = |\chi|$, the constraint enforces the flow to be close to the weighted median of the flow vectors in the window W [14]; whereas if $\psi(\chi) = \chi^2$, the constraint regularizes the flow by the weighted mean [13]. The weight functions ω_1 and ω_2 act as soft segmentation operators, by weighting down the possible outliers in the window. Significant improvement has been reported [13], [14], but the weights either have to be stored, incurring a large memory requirement, or computed in each optimization iteration, which increases computational expense.

We now show that a hard segmentation by the edge can significantly improve the efficiency of these methods. In the $\zeta\eta$-frame, the edge axis partitions W into two subregions W_a and W_b. Note that the two subregions are located in the positive and negative sides of the η axis exclusively. In other words, $W_a = \{(\zeta,\eta)|\eta > 0\}$ and $W_b = \{(\zeta,\eta)|\eta < 0\}$, as shown in Fig. 1.

If the current pixel p is on a motion boundary, it is very likely that the intensity edge is where the two individually moving objects meet. This means that one of the subregions contains mainly the motion inliers of p, and the other contains mainly the motion outliers. To recognize the inlier subregion, we compute the average intensity value $\bar{E}_a$ (resp. $\bar{E}_b$) of W_a (resp. W_b) and compare them to p's intensity $E(p)$. The inlier region W_{in} is determined as

$$W_{\text{in}} = \begin{cases} W_a, & \text{if } \left|\bar{E}_a - E(p)\right| < \left|\bar{E}_b - E(p)\right| \\ W_b, & \text{otherwise.} \end{cases} \quad (16)$$

Based on the selection, we model the nonlocal smoothness constraints in the $\zeta\eta$-frame by

$$\begin{aligned} \sum_{(\tilde{\zeta},\tilde{\eta})\in W_{\text{in}}} \psi\left(u(\zeta,\eta) - u(\tilde{\zeta},\tilde{\eta})\right) &= 0, \\ \sum_{(\tilde{\zeta},\tilde{\eta})\in W_{\text{in}}} \psi\left(v(\zeta,\eta) - v(\tilde{\zeta},\tilde{\eta})\right) &= 0 \end{aligned} \quad (17)$$

and define the associated regularization energy term by

$$\mathfrak{E}_{nl} = \sum_{(\tilde{\zeta},\tilde{\eta})\in W_{\text{in}}} \psi\left(u(\zeta,\eta) - u(\tilde{\zeta},\tilde{\eta})\right) + \sum_{(\tilde{\zeta},\tilde{\eta})\in W_{\text{in}}} \psi\left(v(\zeta,\eta) - v(\tilde{\zeta},\tilde{\eta})\right). \quad (18)$$

Selecting W_{in} from W_a and W_b performs a hard segmentation, i.e., the result of which can be stored as an array of binary labels in the flow initialization step. In further computation, the labels are reused to discard the outliers by filtering out the pixels whose η coordinate have the "wrong" sign. This avoids repeatedly computing the weights in previous nonlocal smoothness constraints and hence saves the computation cost.

The energy terms specified by (3), (8), (12), and (18) are integrated into the following cost function:

$$\mathfrak{E} = \iint_{\Omega} \left(\lambda_1 \mathfrak{E}_{\dot{E}} + \lambda_2(\mathfrak{E}_{\dot{E}_{\boldsymbol{d}}} + \mathfrak{E}_{\dot{E}_{\boldsymbol{n}}}) + \lambda_3 \mathfrak{E}_s + \lambda_4 \mathfrak{E}_{nl}\right) d\zeta d\eta \quad (19)$$

where constant λ_i values balance the contribution of each energy term.

We present the numerical optimization scheme of this objective function in Section IV. This scheme relies on accurate calculation of local intrinsic directions at each pixel, which is the subject of the next section.

III. Intrinsic Direction Estimation and the SO Coordinate Frame

Many previous works detect the edge direction by $\boldsymbol{d} = (\nabla E^{\perp}/\|\nabla E\|)$ and the normal direction by $\boldsymbol{n} = (\nabla E/\|\nabla E)$ (e.g., in [8]), due to the derivation that

$$E_{\boldsymbol{n}} = \langle \nabla E, \boldsymbol{n}\rangle = \|\nabla E\|, \quad E_{\boldsymbol{d}} = \langle \nabla E, \boldsymbol{d}\rangle = 0 \quad (20)$$

which leads to the conclusion that $\boldsymbol{n}$ is the direction of maximum change, and $\boldsymbol{d}$ is the isophote direction. However, $E_{\boldsymbol{d}} = \langle \nabla E, \boldsymbol{d}\rangle$ is not necessarily true when E is not continuously differentiable. At the presence of an edge, E behaves like a smooth and differentiable function in the edge direction but a step and discontinuous function in the horizontal or vertical direction (unless the edge is horizontal or vertical). Thus $E_{\boldsymbol{d}}$ exists, but E_x or E_y may not. However, in practice, E_x and E_y are enforced to be the horizontal and vertical finite differencing values, which can be arbitrarily large, and do not necessarily reflect $E_{\boldsymbol{d}}$. A similar problem exists if the directional variation of the flow field is computed by projecting the flow gradient, e.g., setting $u_{\boldsymbol{d}} = \langle \nabla u, \boldsymbol{d}\rangle$, which is widely adopted in previous works.

Fig. 2 shows two examples where the horizontal–vertical finite differencing fails to predict $\boldsymbol{d}$ or $u_{\boldsymbol{d}}$. Fig. 2(a) depicts a discrete binary image of a black diagonal line lying against

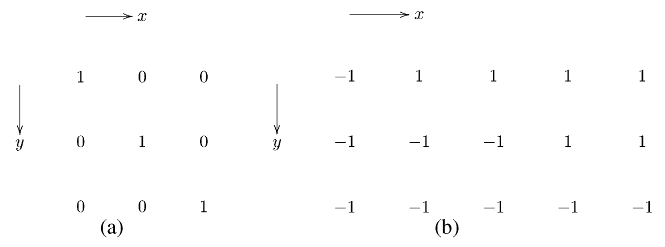

Fig. 2. Example situations where the intensity or the motion field is nondifferentiable. Finite differencing leads to the wrong estimation of the gradient and, consequently, the wrong computation of the edge direction and the directional variation. (a) Discrete image of a black line lying against a white background. The binary numbers indicate the intensity values of image pixels. The intensity function of the center pixel is not differentiable in the horizontal and vertical directions but differentiable in the diagonal direction. The edge direction cannot be estimated accurately by the gradient method because $\nabla E^{\perp}$ is erroneously approximated by the finite differencing (either forward or central) scheme. (b) Discrete motion field of two objects moving to each other with opposite motions $u = 1$ and $u = -1$. The numbers indicate the motion values of image pixels. u is not differentiable in the horizontal and vertical directions but differentiable in the direction $\boldsymbol{d} = [2, 1]/\sqrt{5}$. Projecting the approximated (either by forward or central differencing) ∇u to this direction overestimates the directional variation. One can easily construct similar examples for backward differencing.

a white background. The center pixel of the image is nondifferentiable in the x- and y-directions. Consequently, approximating E_x and E_y by forward differencing yields $\nabla E^{\perp} = [1, -1]^T/\sqrt{2}$, which is obviously an incorrect estimate of $\boldsymbol{d}$. Fig. 2(b) depicts the discrete motion field of two objects moving toward each other with $u = 1$ and $u = -1$. Again, at the center pixel, u is nondifferentiable in the x- and y-directions. Forward differencing leads to $\nabla u = [2, 0]^T$; subsequently, the projection of ∇u to the direction $\boldsymbol{d} = [2, 1]/\sqrt{5}$ yields $u_{\boldsymbol{d}} = 4/\sqrt{5}$. However, u is a constant function in the edge direction and $u_{\boldsymbol{d}}$ should be zero. One can verify that central differencing also produces a wrong estimate in these two examples and can construct similar counter examples if backward differencing is employed.

This aliasing can be alleviated by the structure tensor method (e.g., [9]) as Gaussian convolution can smooth out the nondifferentiability. However, it may also cause oversmoothing. In this paper, instead of relying the detection of $\boldsymbol{d}$ on the horizontal–vertical finite differencing, we employ the original definition of directional derivatives, i.e.,

$$\frac{\partial E}{\partial \boldsymbol{e}} = \lim_{\tau \to 0^+} \frac{E(X + \tau \boldsymbol{e}, t) - E(X, t)}{\tau}, \tau \in \mathbb{R} \qquad (21)$$

where $X = [x, y]$ and $\boldsymbol{e} = [\cos\theta, \sin\theta]$ is a unit vector with an orientation of θ degrees.

Briefly, we estimate the edge direction approximately by one of the 40 directions that quantize the 2-D plane evenly. The reason for choosing 40 directions is due to the success of the SIFT feature, which uses 36 histogram bins to cover the 360° range of orientation. Moreover, our empirical study also validates the adequacy of using 40 directions. From these quantization directions, we select the direction that has the least intensity curvature (in the magnitude sense), which indicates that the intensity in this direction has a linear pattern. Note that the least intensity variation direction generally has the least curvature. Although the edge direction can be alternatively estimated by the quantization direction that has the least first-order variation, we observe that the least curvature-based detection is more stable. The curvature, which is measured by the second-order directional derivative, is given by the following central differencing scheme:

$$\frac{\partial^2 E}{\partial \boldsymbol{e}^2} \approx E(X + \boldsymbol{e}, t) - 2E(X, t) + E(X - \boldsymbol{e}, t). \qquad (22)$$

If $E(X + \boldsymbol{e}, t)$ and $E(X - \boldsymbol{e}, t)$ are obtained by bilinear interpolation, the differencing can be quickly and easily implemented by convolving the image with a high-pass kernel $\boldsymbol{K}_{(\partial^2 E/\partial \boldsymbol{e}^2)}$. For example, in the case that $\theta \leq (\pi/2)$, we have

$$\boldsymbol{K}_{\frac{\partial^2 E}{\partial \boldsymbol{e}^2}} = \begin{bmatrix} \cos\theta\sin\theta & (1-\cos\theta)\sin\theta & 0 \\ \cos\theta(1-\sin\theta) & -2\left[\begin{smallmatrix}\cos\theta+\sin\theta \\ -\cos\theta\sin\theta\end{smallmatrix}\right] & \cos\theta(1-\sin\theta) \\ 0 & (1-\cos\theta)\sin\theta & \cos\theta\sin\theta \end{bmatrix}.$$

Due to the symmetry of central differencing, the curvatures in directions $\boldsymbol{e} = [\cos\theta, \sin\theta]$ and $-\boldsymbol{e} = [\cos(\pi + \theta), \sin(\pi + \theta)]$ have the same magnitude. Therefore, the direction selection can be limited to the range $[0, \pi)$. This means that only 20 convolutions are needed to determine the locally dominant direction.

By applying this edge direction estimation approach to the example shown by Fig. 2(a), on which the gradient method fails, it can be easily verified that the least directional curvature is reached in the direction $\boldsymbol{e} = [\cos(\pi/4), \sin(\pi/4)]$, which coincides with the true edge direction.

The above directional convolution involves the pixel's immediate neighbors only; hence, we call it pixelwise directional convolution (PDC). To improve the robustness, a Gaussian convolution with the Gaussian kernel G_{σ}, where σ is the standard deviation, can be further applied. Similar to the derivative of Gaussian (DoG), the PDC and Gaussian smoothing can be combined to one convolution, i.e., the directional DoG. As it involves a local region around the pixel, we call it regionwise directional convolution (RDC).

The local SO reference frame is thus constructed in such a way that its axes are always aligned with the detected local edge and normal directions at the current pixel, with the unit length defined as same as in the grid-based reference frame.

IV. Numerical Scheme for Oriented Flow Computation

Having described the calculation of intrinsic directions, we now return to the optimization of the objective function $\mathfrak{E}$ [see (19)]. In this paper, the minimization of $\mathfrak{E}$ follows the optimization routine employed in [2]. To facilitate understanding, this section also adopts the notation system of [2]. However, the optimization of $\mathfrak{E}$ presented here is not a straightforward extension from [2]. First, the $\zeta\eta$ coordinates and the deformation parameters give rise to additional equations and terms that do not appear in the Euler–Lagrange systems of previous works. Second, a traditional numerical discretization scheme in the xy coordinate frame is not applicable to the $\zeta\eta$ frame.

The following settings are in common with [2]. The energy functional $\mathfrak{E}$ is minimized in the traditional framework of multilevel multistage refinement. In particular, an L-level Gaussian pyramid is constructed by recursively subsampling the original sequence. In each level l, the flow is refined by successive stages of image warping, i.e., at each stage $s = 0, \ldots, S$, the flow increment $\delta u^{l,s}$, $\delta v^{l,s}$ is obtained by solving the Euler–Lagrange equations associated with $\mathfrak{E}^{l,s}$. The flow of stage $s+1$ is updated by $(u^{l,s+1}, v^{l,s+1}) = (u^{l,s}, v^{l,s}) + (\delta u^{l,s}, \delta v^{l,s})$. After S stages of refinement, the flow is propagated to level $l-1$ by $(u^{l-1,0}, v^{l-1,0}) = k(u^{l,S}, v^{l,S})$, where k is the sampling factor. The penalizing function is specified by $\psi(\chi) = \sqrt{\chi^2 + \epsilon}$, the objective functional is minimized by solving the associated Euler–Lagrange equation system. To obtain the numerical solutions by linear schemes such as the Jacobi or Gauss–Seidel iteration, an inner iteration is used to remove the nonlinearity introduced by $\psi(\chi)$. The derivation below focuses on simplifying the Euler–Lagrange system and the discretization scheme that leads to convergence in the $\zeta\eta$ frame.

In this coarse-to-fine optimization structure, the core of the algorithm is thus to minimize

$$\mathfrak{E}^{l,s} = \iint\limits_{\Omega^{l,s}} \left(\lambda_1 \mathfrak{E}^{l,s}_{\dot{E}} + \lambda_2\left(\mathfrak{E}^{l,s}_{\dot{E}_{\boldsymbol{d}}} + \mathfrak{E}^{l,s}_{\dot{E}_{\boldsymbol{n}}}\right) + \lambda_3 \mathfrak{E}^{l,s}_{s} + \lambda_4 \mathfrak{E}^{l,s}_{nl}\right) d\zeta d\eta \tag{23}$$

where the superscripts index the optimization context. We approximate the data terms by the chain rule

$$\begin{aligned}
\mathfrak{E}^{l,s}_{\dot{E}} &= \psi(\dot{E}^{l,s}) \\
&\approx \psi\left(E^{l,s}_x \delta u^{l,s} + E^{l,s}_y \delta v^{l,s} + E^{l,s}_t\right) \\
\mathfrak{E}^{l,s}_{\dot{E}_{\boldsymbol{d}}} &= \psi\left(\dot{E}^{l,s}_{\boldsymbol{d}}\right) \\
&\approx \psi\left(E^{l,s}_{x\boldsymbol{d}} \delta u^{l,s} + E^{l,s}_x \delta u^{l,s}_{\boldsymbol{d}} + E^{l,s}_{y\boldsymbol{d}} \delta v^{l,s} + E^{l,s}_y \delta v^{l,s}_{\boldsymbol{d}} + E^{l,s}_{t\boldsymbol{d}}\right) \\
\mathfrak{E}^{l,s}_{\dot{E}_{\boldsymbol{n}}} &= \psi\left(\dot{E}^{l,s}_{\boldsymbol{n}}\right) \\
&\approx \psi\left(E^{l,s}_{x\boldsymbol{n}} \delta u^{l,s} + E^{l,s}_x \delta u^{l,s}_{\boldsymbol{n}} + E^{l,s}_{y\boldsymbol{n}} \delta v^{l,s} + E^{l,s}_y \delta v^{l,s}_{\boldsymbol{n}} + E^{l,s}_{t\boldsymbol{n}}\right)
\end{aligned} \tag{24}$$

with the intensity partial derivatives estimated from the first image and the warped second image. The local smoothness term of $\mathfrak{E}^{l,s}$ reads

$$\mathfrak{E}^{l,s}_s = \psi\left(\left\|\boldsymbol{m}^{l,s}_{\boldsymbol{d}} + \delta \boldsymbol{m}^{l,s}_{\boldsymbol{d}}\right\|_2\right) + \psi\left(\left\|\boldsymbol{m}^{l,s}_{\boldsymbol{n}} + \delta \boldsymbol{m}^{l,s}_{\boldsymbol{n}}\right\|_2\right) \tag{25}$$

where $\delta\boldsymbol{m}^{l,s}$ is the increment of $\boldsymbol{m}$ in level l and stage s.

The minimizer of $\mathfrak{E}^{l,s}$ fulfills the Euler–Lagrange system as follows:

$$\frac{\partial \mathfrak{E}^{l,s}}{\partial \delta \boldsymbol{m}^{l,s}} - \frac{\partial}{\partial \boldsymbol{d}}\left(\frac{\partial \mathfrak{E}^{l,s}}{\partial \delta \boldsymbol{m}^{l,s}_{\boldsymbol{d}}}\right) - \frac{\partial}{\partial \boldsymbol{n}}\left(\frac{\partial \mathfrak{E}^{l,s}}{\partial \delta \boldsymbol{m}^{l,s}_{\boldsymbol{n}}}\right) = 0 \tag{26}$$

which consists of six equations, with the last four equations arising from the deformation fields $\delta u^{l,s}_{\boldsymbol{d}}$, $\delta v^{l,s}_{\boldsymbol{d}}$, $\delta u^{l,s}_{\boldsymbol{n}}$, and $\delta v^{l,s}_{\boldsymbol{n}}$. Moreover, by defining $\rho(\chi) = (\partial\psi(\chi)/\partial\chi)$, which can be further simplified to $\rho(\chi) = (\chi/\psi(\chi))$, the first two equations of (26) read

$$\begin{aligned}
&\lambda_1 E^{l,s}_x \rho(\dot{E}^{l,s}) + \lambda_2 \begin{pmatrix} E^{l,s}_{x\boldsymbol{d}} & E^{l,s}_{x\boldsymbol{n}} \end{pmatrix} \begin{pmatrix} \rho\left(\dot{E}^{l,s}_{\boldsymbol{d}}\right) \\ \rho\left(\dot{E}^{l,s}_{\boldsymbol{n}}\right) \end{pmatrix} \\
&- \lambda_2 E^{l,s}_x \mathrm{div}_d \begin{pmatrix} \rho\left(\dot{E}^{l,s}_{\boldsymbol{d}}\right) \\ \rho\left(\dot{E}^{l,s}_{\boldsymbol{n}}\right) \end{pmatrix} - \lambda_3 \mathrm{div}_d \begin{pmatrix} \frac{u^{l,s}_{\boldsymbol{d}} + \delta u^{l,s}_{\boldsymbol{d}}}{\psi(\|\boldsymbol{m}^{l,s}_{\boldsymbol{d}} + \delta \boldsymbol{m}^{l,s}_{\boldsymbol{d}}\|_2)} \\ \frac{u^{l,s}_{\boldsymbol{n}} + \delta u^{l,s}_{\boldsymbol{n}}}{\psi(\|\boldsymbol{m}^{l,s}_{\boldsymbol{n}} + \delta \boldsymbol{m}^{l,s}_{\boldsymbol{n}}\|_2)} \end{pmatrix} \\
&- \lambda_4 \sum_{i \in W_{\text{in}}} \frac{u^{l,s} + \delta u^{l,s} - u^{l,s}_i - \delta u^{l,s}_i}{\psi\left(u^{l,s} + \delta u^{l,s} - u^{l,s}_i - \delta u^{l,s}_i\right)} = 0,
\end{aligned} \tag{27}$$

$$\begin{aligned}
&\lambda_1 E^{l,s}_y \rho(\dot{E}^{l,s}) + \lambda_2 \begin{pmatrix} E^{l,s}_{y\boldsymbol{d}} & E^{l,s}_{y\boldsymbol{n}} \end{pmatrix} \begin{pmatrix} \rho\left(\dot{E}^{l,s}_{\boldsymbol{d}}\right) \\ \rho\left(\dot{E}^{l,s}_{\boldsymbol{n}}\right) \end{pmatrix} \\
&- \lambda_2 E^{l,s}_y \mathrm{div}_d \begin{pmatrix} \rho\left(\dot{E}^{l,s}_{\boldsymbol{d}}\right) \\ \rho\left(\dot{E}^{l,s}_{\boldsymbol{n}}\right) \end{pmatrix} - \lambda_3 \mathrm{div}_d \begin{pmatrix} \frac{v^{l,s}_{\boldsymbol{d}} + \delta v^{l,s}_{\boldsymbol{d}}}{\psi(\|\boldsymbol{m}^{l,s}_{\boldsymbol{d}} + \delta \boldsymbol{m}^{l,s}_{\boldsymbol{d}}\|_2)} \\ \frac{v^{l,s}_{\boldsymbol{n}} + \delta v^{l,s}_{\boldsymbol{n}}}{\psi(\|\boldsymbol{m}^{l,s}_{\boldsymbol{n}} + \delta \boldsymbol{m}^{l,s}_{\boldsymbol{n}}\|_2)} \end{pmatrix}
\end{aligned} \tag{28}$$

$$- \lambda_4 \sum_{i \in W_{\text{in}}} \frac{v^{l,s} + \delta v^{l,s} - v^{l,s}_i - \delta v^{l,s}_i}{\psi\left(v^{l,s} + \delta v^{l,s} - v^{l,s}_i - \delta v^{l,s}_i\right)} = 0 \tag{29}$$

where the divergence notation div_d in the $\zeta\eta$-frame means

$$\mathrm{div}_d \begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} = \frac{\partial \chi_1}{\partial \boldsymbol{d}} + \frac{\partial \chi_2}{\partial \boldsymbol{n}}.$$

It can be seen that these deformation fields also appear in $\rho(\dot{E}^{l,s}_{\boldsymbol{d}})$ and $\rho(\dot{E}^{l,s}_{\boldsymbol{n}})$ of (29). These additional equations and terms do not appear in previous works as previous objective functionals are generally formulated by two fields, i.e., $\delta u^{l,s}$ and $\delta v^{l,s}$, rather than an affine motion model. Consequently, previous Euler–Lagrange systems generally have two equations with two unknown fields.

Although the affine motion model is adaptive and is expected to improve the flow computation accuracy, solving a system of six equations substantially increases the computation load compared with previous works. Inspired by the nonlinearity removal strategy of Brox *et al.* [2], we decouple the deformation functions (assembled by $\alpha^{l,s}$) from (29) along with the linearization.

The nonlinearity of (29) is caused by the nonlinearity of function $\rho(\chi)$. Brox *et al.* propose using an inner iteration to remove the nonlinearity. Let r index the inner iteration. The flow increment is initialized by $(\delta u^{l,s,r=0}, \delta v^{l,s,r=0}) = (\delta u^{l,s}, \delta v^{l,s})$. In the $(r+1)$th iteration, $\rho(\chi^{l,s,r+1}) = (\chi^{l,s,r+1}/\psi(\chi^{l,s,r+1}))$ is approximated by $\rho(\chi^{l,s,r+1}) \approx (\chi^{l,s,r+1}/\psi(\chi^{l,s,r}))$, which becomes a linear function of $\chi^{l,s,r+1}$.

In the same vein, we decouple $\alpha^{l,s,r+1}$ from the computation of $\delta u^{l,s,r+1}$ and $\delta v^{l,s,r+1}$ by replacing $\alpha^{l,s,r+1}$ with $\alpha^{l,s,r}$. In particular, in the rth iteration, after $\delta u^{l,s,r}$ and $\delta v^{l,s,r}$ are obtained, we derive $\alpha^{l,s,r}$ from $\delta u^{l,s,r}$ and $\delta v^{l,s,r}$ by the definition of directional derivatives, instead of solving the last four equations of (26). This simplifies the Euler–Lagrange equation system to two equations. In addition, in the $(r+1)$th iteration, the involvement of $\alpha^{l,s,r+1}$ is removed by the following approximation scheme:

- in the case that $\rho(\dot{E}^{l,s,\mathbf{r+1}}_{\vec{\boldsymbol{d}}})$ and $\rho(\dot{E}^{l,s,\mathbf{r+1}}_{\vec{\boldsymbol{n}}})$ appear in the divergence term of (29), i.e., they will be further differentiated with respect to $\vec{\boldsymbol{d}}$ and $\vec{\boldsymbol{n}}$, they are approximated as shown at the bottom of the next page.
- in the case that $\rho(\dot{E}^{l,s,\mathbf{r+1}}_{\boldsymbol{d}})$ and $\rho(\dot{E}^{l,s,\mathbf{r+1}}_{\boldsymbol{n}})$ will not be further differentiated, they are approximated as shown at the bottom of the next page.

The term $\rho(\dot{E}^{l,s,\mathbf{r+1}})$ in the same iteration is simply approximated by

$$\rho\left(\dot{E}^{l,s,\mathbf{r+1}}\right) \approx \frac{E_x^{l,s}\delta u^{l,s,\mathbf{r+1}} + E_y^{l,s}\delta v^{l,s,\mathbf{r+1}} + E_t^{l,s}}{\psi\left(E_x^{l,s}\delta u^{l,s,\mathbf{r}} + E_y^{l,s}\delta v^{l,s,\mathbf{r}} + E_t^{l,s}\right)}. \quad (30)$$

The above approximation schemes convert (29) to equations of $\delta u^{l,s,\mathbf{r+1}}$, $\delta u_{\boldsymbol{dd}}^{l,s,\mathbf{r+1}}$, $\delta u_{\boldsymbol{nn}}^{l,s,\mathbf{r+1}}$, $\delta v^{l,s,\mathbf{r+1}}$, $\delta v_{\boldsymbol{dd}}^{l,s,\mathbf{r+1}}$, and $\delta v_{\boldsymbol{nn}}^{l,s,\mathbf{r+1}}$, by replacing $\delta u_{\boldsymbol{d}}^{l,s,\mathbf{r+1}}$, $\delta u_{\boldsymbol{n}}^{l,s,\mathbf{r+1}}$, $\delta v_{\boldsymbol{d}}^{l,s,\mathbf{r+1}}$, and $\delta v_{\boldsymbol{n}}^{l,s,\mathbf{r+1}}$ with $\delta u_{\boldsymbol{d}}^{l,s,\mathbf{r}}$, $\delta u_{\boldsymbol{n}}^{l,s,\mathbf{r}}$, $\delta v_{\boldsymbol{d}}^{l,s,\mathbf{r}}$, and $\delta v_{\boldsymbol{n}}^{l,s,\mathbf{r}}$. This decouples the deformation parameters from the Euler–Lagrange equations of the $(r+1)$th iteration. Now, solving (29) is simplified to iteratively solving the following equation pair:

$$\begin{aligned}
0 =\;& \lambda_1\rho(\dot{E}^{l,s,r+1})E_x^{l,s} \\
&+ \lambda_2\left(\rho\left(\dot{E}_{\boldsymbol{d}}^{l,s,r+1}\right)E_{x\boldsymbol{d}}^{l,s} + \rho\left(\dot{E}_{\boldsymbol{n}}^{l,s,r+1}\right)E_{x\boldsymbol{n}}^{l,s}\right) \\
&- \lambda_2 E_x^{l,s}\mathrm{div}_d\begin{pmatrix}\rho\left(\dot{E}_{\boldsymbol{d}}^{l,s,r+1}\right)\\ \rho\left(\dot{E}_{\boldsymbol{n}}^{l,s,r+1}\right)\end{pmatrix} \\
&- \lambda_3\mathrm{div}_d\begin{pmatrix}\frac{u_{\boldsymbol{d}}^{l,s}+\delta u_{\boldsymbol{d}}^{l,s,r+1}}{\psi\left(\left\|\boldsymbol{m}_{\boldsymbol{d}}^{l,s}+\delta\boldsymbol{m}_{\boldsymbol{d}}^{l,s,r}\right\|_2\right)}\\ \frac{u_{\boldsymbol{n}}^{l,s}+\delta u_{\boldsymbol{n}}^{l,s,r+1}}{\psi\left(\left\|\boldsymbol{m}_{\boldsymbol{n}}^{l,s}+\delta\boldsymbol{m}_{\boldsymbol{n}}^{l,s,r}\right\|_2\right)}\end{pmatrix} \\
&- \lambda_4\sum_{i\in W_{\mathrm{in}}}\frac{u^{l,s}+\delta u^{l,s,r+1}-u_i^{l,s}-\delta u_i^{l,s,r+1}}{\psi\left(u^{l,s}+\delta u^{l,s,r}-u_i^{l,s}-\delta u_i^{l,s,r}\right)} \\
0 =\;& \lambda_1\rho(\dot{E}^{l,s,r+1})E_y^{l,s} \\
&+ \lambda_2\left(\rho\left(\dot{E}_{\boldsymbol{d}}^{l,s,r+1}\right)E_{y\boldsymbol{d}}^{l,s} + \rho\left(\dot{E}_{\boldsymbol{n}}^{l,s,r+1}\right)E_{y\boldsymbol{n}}^{l,s}\right) \\
&- \lambda_2 E_y^{l,s}\mathrm{div}_d\begin{pmatrix}\rho\left(\dot{E}_{\boldsymbol{d}}^{l,s,r+1}\right)\\ \rho\left(\dot{E}_{\boldsymbol{n}}^{l,s,r+1}\right)\end{pmatrix} \\
&- \lambda_3\mathrm{div}_d\begin{pmatrix}\frac{v_{\boldsymbol{d}}^{l,s}+\delta v_{\boldsymbol{d}}^{l,s,r+1}}{\psi\left(\left\|\boldsymbol{m}_{\boldsymbol{d}}^{l,s}+\delta\boldsymbol{m}_{\boldsymbol{d}}^{l,s,r}\right\|_2\right)}\\ \frac{v_{\boldsymbol{n}}^{l,s}+\delta v_{\boldsymbol{n}}^{l,s,r+1}}{\psi\left(\left\|\boldsymbol{m}_{\boldsymbol{n}}^{l,s}+\delta\boldsymbol{m}_{\boldsymbol{n}}^{l,s,r}\right\|_2\right)}\end{pmatrix} \\
&- \lambda_4\sum_{i\in W_{\mathrm{in}}}\frac{v^{l,s}+\delta v^{l,s,r+1}-v_i^{l,s}-\delta v_i^{l,s,r+1}}{\psi\left(v^{l,s}+\delta v^{l,s,r}-v_i^{l,s}-\delta v_i^{l,s,r}\right)}. \quad (31)
\end{aligned}$$

To obtain the numerical solution of (31), we discretize the divergence terms by the following energy-preserving central differencing:

$$\begin{aligned}
&\mathrm{div}_d\begin{pmatrix}p(\zeta,\eta)\delta u_{\boldsymbol{d}}^{l,s,r+1}\\ q(\zeta,\eta)\delta u_{\boldsymbol{n}}^{l,s,r+1}\end{pmatrix} \\
&= p\left(\zeta+\frac{1}{2},\eta\right)\left(\delta u^{l,s,r+1}(\zeta+1,\eta)-\delta u^{l,s,r+1}(\zeta,\eta)\right) \\
&\quad + p\left(\zeta-\frac{1}{2},\eta\right)\left(\delta u^{l,s,r+1}(\zeta-1,\eta)-\delta u^{l,s,r+1}(\zeta,\eta)\right) \\
&\quad + q\left(\zeta,\eta+\frac{1}{2}\right)\left(\delta u^{l,s,r+1}(\zeta,\eta+1)-\delta u^{l,s,r+1}(\zeta,\eta)\right) \\
&\quad + q\left(\zeta,\eta-\frac{1}{2}\right)\left(\delta u^{l,s,r+1}(\zeta,\eta-1)-\delta u^{l,s,r+1}(\zeta,\eta)\right)
\end{aligned} \quad (32)$$

Now, the Euler–Lagrange equation system is simplified to equations of $\delta u^{l,s,r+1}$ and $\delta v^{l,s,r+1}$.

Let $\mathcal{N} = \{\zeta\pm 1, \eta\pm 1\}$ collect the four immediate neighbors of pixel (ζ,η) in the $\zeta\eta$ coordinates. With the differencing scheme (32), (31) can be rewritten into the numerical form

$$\begin{aligned}
\delta u^{l,s,r+1}(\zeta,\eta) =\;& \omega\delta v^{l,s,r+1}(\zeta,\eta) \\
&+ \sum_{j\in\mathcal{N}}\beta_j\delta u_j^{l,s,r+1} + \sum_{j\in\mathcal{N}}\xi_j\delta v_j^{l,s,r+1} \\
&+ \sum_{i\in W_{\mathrm{in}}}\delta u_i^{l,s,r+1} + C_1 \\
\delta v^{l,s,r+1}(\zeta,\eta) =\;& \varpi\delta u^{l,s,r+1}(\zeta,\eta) \\
&+ \sum_{j\in\mathcal{N}}\kappa_j\delta u_j^{l,s,r+1} + \sum_{j\in\mathcal{N}}\gamma_j\delta v_j^{l,s,r+1} \\
&+ \sum_{i\in W_{\mathrm{in}}}\delta v_i^{l,s,r+1} + C_2 \quad (33)
\end{aligned}$$

where ω, ϖ, β_j, κ_j, γ_j, ξ_j, C_1, and C_2 are functions of the partial derivatives of E and motion deformation components $\alpha^{l,s,r}$ and are constant with respect to fixed l, s, and r.

The above equation pair demonstrates another difference between this paper and previous works: the flow of a pixel is

---

$$\begin{aligned}
\rho\left(\dot{E}_{\boldsymbol{d}}^{l,s,\mathbf{r+1}}\right) &\approx \frac{E_{x\boldsymbol{d}}^{l,s}\delta u^{l,s,\mathbf{r}} + E_x^{l,s}\delta u_{\boldsymbol{d}}^{l,s,\mathbf{r+1}} + E_{y\boldsymbol{d}}^{l,s}\delta v^{l,s,\mathbf{r}} + E_y^{l,s}\delta v_{\boldsymbol{d}}^{l,s,\mathbf{r+1}} + E_{t\boldsymbol{d}}^{l,s}}{\psi\left(E_{x\boldsymbol{d}}^{l,s}\delta u^{l,s,\mathbf{r}} + E_x^{l,s}\delta u_{\boldsymbol{d}}^{l,s,\mathbf{r}} + E_{y\boldsymbol{d}}^{l,s}\delta v^{l,s,\mathbf{r}} + E_y^{l,s}\delta v_{\boldsymbol{d}}^{l,s,\mathbf{r}} + E_{t\boldsymbol{d}}^{l,s}\right)} \\
\rho\left(\dot{E}_{\boldsymbol{n}}^{l,s,\mathbf{r+1}}\right) &\approx \frac{E_{x\boldsymbol{n}}^{l,s}\delta u^{l,s,\mathbf{r}} + E_x^{l,s}\delta u_{\boldsymbol{n}}^{l,s,\mathbf{r+1}} + E_{y\boldsymbol{n}}^{l,s}\delta v^{l,s,\mathbf{r}} + E_y^{l,s}\delta v_{\boldsymbol{n}}^{l,s,\mathbf{r+1}} + E_{t\boldsymbol{n}}^{l,s}}{\psi\left(E_{x\boldsymbol{n}}^{l,s}\delta u^{l,s,\mathbf{r}} + E_x^{l,s}\delta u_{\boldsymbol{n}}^{l,s,\mathbf{r}} + E_{y\boldsymbol{n}}^{l,s}\delta v^{l,s,\mathbf{r}} + E_y^{l,s}\delta v_{\boldsymbol{n}}^{l,s,\mathbf{r}} + E_{t\boldsymbol{n}}^{l,s}\right)}
\end{aligned}$$

$$\begin{aligned}
\rho\left(\dot{E}_{\boldsymbol{d}}^{l,s,\mathbf{r+1}}\right) &\approx \frac{E_{x\boldsymbol{d}}^{l,s}\delta u^{l,s,\mathbf{r+1}} + E_x^{l,s}\delta u_{\boldsymbol{d}}^{l,s,\mathbf{r}} + E_{y\boldsymbol{d}}^{l,s}\delta v^{l,s,\mathbf{r+1}} + E_y^{l,s}\delta v_{\boldsymbol{d}}^{l,s,\mathbf{r}} + E_{t\boldsymbol{d}}^{l,s}}{\psi\left(E_{x\boldsymbol{d}}^{l,s}\delta u^{l,s,\mathbf{r}} + E_x^{l,s}\delta u_{\boldsymbol{d}}^{l,s,\mathbf{r}} + E_{y\boldsymbol{d}}^{l,s}\delta v^{l,s,\mathbf{r}} + E_y^{l,s}\delta v_{\boldsymbol{d}}^{l,s,\mathbf{r}} + E_{t\boldsymbol{d}}^{l,s}\right)} \\
\rho\left(\dot{E}_{\boldsymbol{n}}^{l,s,\mathbf{r+1}}\right) &\approx \frac{E_{x\boldsymbol{n}}^{l,s}\delta u^{l,s,\mathbf{r+1}} + E_x^{l,s}\delta u_{\boldsymbol{n}}^{l,s,\mathbf{r}} + E_{y\boldsymbol{n}}^{l,s}\delta v^{l,s,\mathbf{r+1}} + E_y^{l,s}\delta v_{\boldsymbol{n}}^{l,s,\mathbf{r}} + E_{t\boldsymbol{n}}^{l,s}}{\psi\left(E_{x\boldsymbol{n}}^{l,s}\delta u^{l,s,\mathbf{r}} + E_x^{l,s}\delta u_{\boldsymbol{n}}^{l,s,\mathbf{r}} + E_{y\boldsymbol{n}}^{l,s}\delta v^{l,s,\mathbf{r}} + E_y^{l,s}\delta v_{\boldsymbol{n}}^{l,s,\mathbf{r}} + E_{t\boldsymbol{n}}^{l,s}\right)}
\end{aligned}$$

"diffused" from its four immediate neighbors in the edge and normal directions, rather than the horizontal and vertical directions. Hence, the local structure information is embedded into the flow field. A numerical scheme such as the Jacobi iteration or the Gauss–Seidel iteration straightforwardly recovers $\delta u^{l,s,r+1}$ and $\delta v^{l,s,r+1}$ from (33).

In summary, the optical flow is recovered by nested loops of iterations. As pointed out by Brox *et al.* [2], the multilevel multistage minimization of $\mathfrak{E}^{l,s}$ performs a fixed point iteration to minimize $\mathfrak{E}$. Due to the convexity of $\mathfrak{E}^{l,s}$, there exits a unique global minimum, which satisfies the associated Euler–Lagrange equations. However, the nonlinearity-removal iteration (indexed by r), which is in a very similar spirit to iterative reweighted least squares for L1-minimization [28], is not guaranteed to converge to the global minimum. The convergence of the Jacobi or Gauss–Seidel iteration depends on the formulation of the discretized Euler–Lagrange equations. In all our experiments, 150 Jacobi iterations are enough for (33) to converge.

V. Experimental Results

This paper mainly focuses on SO flow computation (see Section II), and accordingly, the first set of experiments examines the effect of this method on several challenging sequences. In addition, we propose a directional convolution method for edge detection (see Section III), a new scheme to compute directional derivatives (see Section III), and a nonlocal regularization term based on the intrinsic structure-aligned segmentation (see Section II-D). We test each of these proposals using separate experiments, in which we isolate the new technique and compare it to its traditional counterpart.

All the flow computation results presented in this paper are obtained on grayscale sequences of two frames. Following [14], the image pair is recursively subsampled by a factor of 2 to construct the computation pyramid until the height of the subsampled image reaches 20–30 pixels. In each pyramid level, the flow is refined for five stages of image warping. In each stage, five inner iterations are implemented to linearize the nonlinear Euler–Lagrange equations. Parameter λ_i values are set based on the sequence nature: If the underlying motion field is smooth, λ_3 and λ_4 should be large; otherwise, they should be small to preserve motion detail [10]. For this purpose, we design a simple approach to tune the parameters automatically. At the top level, the parameters are initialized by $\lambda_1 = 1$, $\lambda_2 = 5$, and $\lambda_3^L = \lambda_4^L = 1000$. In the subsequent levels $l = L-1, \ldots, 1$, λ_3^l and λ_4^l are adapted according to the estimated motion smoothness. Specifically, we compute the flow variation of level $l+1$ by $\|\nabla u^{l+1}\|_2 + \|\nabla v^{l+1}\|_2$ and then count the portion of pixels whose flow magnitude falls outside three standard deviations from the mean value of the overall distribution. This portion index, which is denoted by P^{l+1}, is observed to be a low-cost indicator of the amount of motion details. In all our experiments, it ranges from 0.005 to 0.03. Therefore, in level l, λ_3, and λ_4 are adjusted to

$$\lambda_3^l = \lambda_4^l = \frac{1000}{\max\left(0.5, \mathrm{round}(100 \times P^{l+1})\right)}. \tag{34}$$

The nonlocal window size is set by default to be 7×7 at the top level. In the subsequent levels, if $P > 0.02$, it remains the same; otherwise, it is increased to 9×9. The flow computation accuracy is measured by conventional metrics of average angular error (AAE) and average endpoint error (AEE) [29].

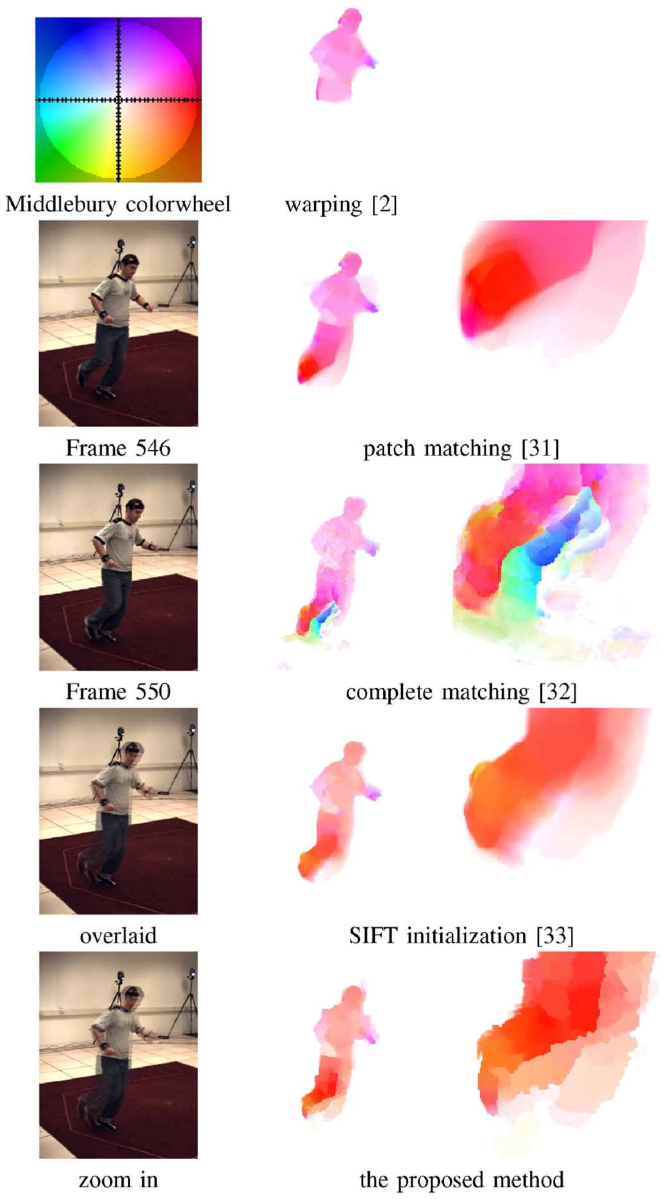

Fig. 3. Results of the comparison conducted on HumanEva-II. Results by previous works are taken from the corresponding publications. As different reference uses different color coding scheme, we adjust the pictures to the same coding scheme, i.e., the Middlebury colorwheel. Compared with techniques such as warping [2] and patch matching [31], the proposed method estimates the motion of the lower body with more accurate direction and magnitude. The result is also competitive with [32] and [33].

A. SO Optical Flow Computation

1) Application to Human Motion Sequences (SO Computation Versus the State of the Art): As the SO flow computation is rotationally robust and discontinuity preserving, it benefits applications such as human motion estimation, which in many situations can be simplified to a series of rotations around translating joints. Fig. 3 shows an example sequence HumanEva-II created by Brown University [30]. This sequence is challenging for optical flow computation because the running person's right

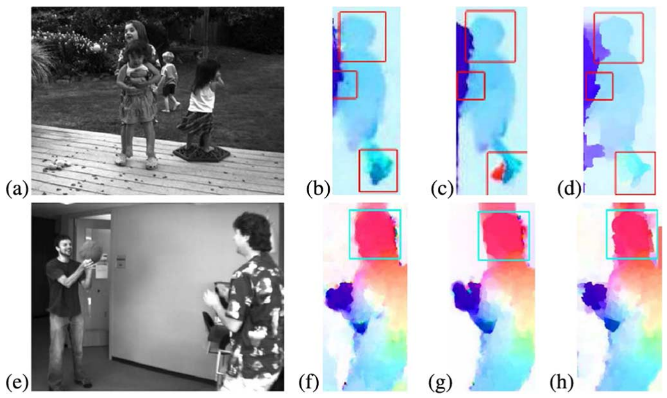

Fig. 4. Optical flow estimated on sequences Backyard and Basketball. Compared with methods that have the top performance on these sequences, the proposed SO computation recovers the motion of the body segments with competitive quality and preserves the profile of the moving persons faithfully (See the highlighted areas). (a) Test frame of sequence Backyard. (b) Optical flow of the walking boy computed by Trobin *et al*'s CBF, which has the least interpolation error on Backyard in the Middlebury evaluation. (c) Flow computed by Wedel *et al*'s F-TV-L1, which achieves the least normalized interpolation error on Backyard in the Middlebury evaluation. (d) Flow computed by the proposed method. (e) Test frame of sequence Basketball. (f) Optical flow of the walking boy computed by Xu *et al*'s MDP-FLOW2, which has the least interpolation error on Basketball. (g) Flow computed by Drulea's CLG-TV, which has the least normalized interpolation error in the Middlebury evaluation. (h) Flow computed by the proposed method. Interested readers are referred to the Middlebury evaluation website for more information.

foot undergoes a large translation and fast rotation around the ankle. The optical flow of this area is about 20 pixels/frame.[4] This sequence has been used in [31]–[33] to test large displacement estimation techniques. To overcome the difficulty, all these works employ matching. For example, [31] uses patch matching to guide the flow computation, [32] optimizes the cost function by a brute-force search in the restricted range, and [33] initializes the flow computation at each pyramid level with the aid of SIFT matching. Although matching improves the accuracy for large displacement computation, it drastically increases the computation complexity.

Compared with previous works, the SO flow computation captures the motion of each body segment and recovers the motion of the foot with the right direction and magnitude. The motion boundary between the foot and the leg is sharp enough to discern the shape of the foot from the recovered flow. This result is comparable with the ones by [32] and [33]; however, the proposed method achieves the result by simply orienting the coordinate system, i.e., the computational cost of which is evidently cheaper than the restricted brute-force search and SIFT matching.

Fig. 4 shows another two examples on the Middlebury sequences Backyard and Basketball. For comparison, Fig. 4 includes the flow results achieved by two methods which achieve the least interpolation error or the normalized interpolation error on these two sequences in the Middlebury evaluation website. Although our estimation of the background motion is inferior to the best-performing methods, our human motion recovery has competitive quality. For example, on sequence backyard, the SO computation method recovers the motion contour of the walking boy clearly. Moreover, the motion boundaries are recovered in the head–neck area and the sleeve–arm area. Although all methods suffer some errors around the moving foot, the shape and position of the foot and leg can be easily recognized from the flow computed by the proposed method. Sequence Basketball also contains challenging human motion. The SO computation and the reference methods estimate the motion of the right persons' arms and hands with similar visual quality, yet the SO computation preserves the profile of the right person more faithfully on estimating the head's motion.

[4]Following previous works to simulate large displacement, we extract frames 546 and 550 as two adjacent frames.

2) Evaluation on Synthesized Sequences ($\zeta\eta$-Frame Versus xy-Frame): We verify the stability, robustness, and generality of the proposed algorithm on the Middlebury test and training sequences. At the time of writing, the SO computation ranks the 21st by AAE and 22nd by AEE among the 56 methods evaluated by the Middlebury testing [34]. Given that our objective functional consists of the most fundamental flow constraints and we use the minimum image information, this evaluation result validates the effectiveness of the locally oriented coordinate frame. Moreover, on the Urban sequence, which consists mainly of structures, our algorithm ranks the third by AAE. This performance is consistent with our attempt to exploit the intrinsic structure for flow computation.

The first two rows of Table I numerically compare the performance of SO computation and grid-based computation (i.e., the degenerated SO computation with $\boldsymbol{d} = [1, 0]^T$ and $\boldsymbol{n} = [0, 1]^T$). Except for the coordinate frames, all settings for the two computation schemes are the same. As there is a lack of fast rotation simulated in these sequences, the results mainly reflect the motion boundary preserving ability of the SO regularization. Noticeable improvement by the SO computation can be observed on sequences Urban2, Urban3, Venus, and RubberWhale. The difference between the performance of the two schemes on sequences Grove3 and Grove2 is less significant. The two schemes achieve almost equivalent accuracy on sequences Hydrangea and Dimetrodon. From these experiments, we observe that the SO computation yields better accuracy on sequences that contain a large amount of large-scale straight boundaries. This is also confirmed by the qualitative comparison on the Middlebury test sequences Urban and Wooden, as shown in Fig. 5. The reason is that the detection of intrinsic directions of such pixels tends to be more accurate, particularly when the frame is subsampled to the top level of the pyramid. On the other hand, in areas that contain a mixture of directional textures, the accurate detection of the intrinsic direction is difficult. This may result in the inferior performance of the SO computation. Fig. 10 demonstrates two examples where the SO computation fails at the junction of three moving objects.

The average performance of using the affine motion model is better than the translational motion model (i.e., assuming that the deformation parameters are zero), as shown by row 3 of Table I. Since the Middlebury training sequences do not contain substantial oblique motion, the performance difference is moderate. Fig. 6 shows the color-coded flow recovery results and the grayscale-coded AAE results obtained by the SO flow computation on eight Middlebury training sequences. The algorithm was programmed in Matlab and executed on a personal computer with 1-GB memory, Intel Core 2 Duo CPU, and 2792-MHz clock speed. The running time on sequence Urban

TABLE I
OPTICAL FLOW RECOVERY RESULTS BY DIFFERENT SCHEMES. BOLD NUMBERS INDICATE BEST PERFORMANCE

	Venus		Urban3		Urban2		RubberWhale		Grove3		Grove2		Hydrangea		Dimetrodon	
	AAE	AEE	AAE	AEE	AAE	AEE	AAE	AEE	AAE	AEE	AAE	AEE	AAE	AEE	AAE	AEE
StructureOriented (SO)	3.81	0.26	3.77	0.47	**2.49**	**0.29**	3.84	0.12	6.43	0.64	2.43	0.17	2.16	0.18	1.82	0.09
GridBased (SO-NoOri)	4.49	0.29	5.80	0.67	2.88	0.37	4.37	0.13	6.63	0.69	2.69	0.19	2.16	0.18	1.82	0.09
SO-NoAffine	3.91	0.27	**3.45**	**0.45**	2.73	0.30	4.13	0.13	**6.40**	0.64	2.47	0.18	2.21	0.18	1.88	0.10
SO-Projection	3.92	**0.26**	3.95	0.51	2.76	0.34	4.12	0.13	6.45	0.64	2.53	0.18	2.16	0.18	1.87	0.09
SO+DataNorm	3.34	0.24	4.04	0.48	2.46	0.30	3.56	0.11	6.59	0.66	2.29	0.16	2.19	0.18	1.84	0.09
SO+DataNorm+PreSmooth	**3.32**	**0.24**	4.32	0.51	2.62	0.32	**3.26**	**0.11**	6.73	0.69	**2.28**	**0.16**	**1.99**	**0.17**	**1.77**	**0.09**

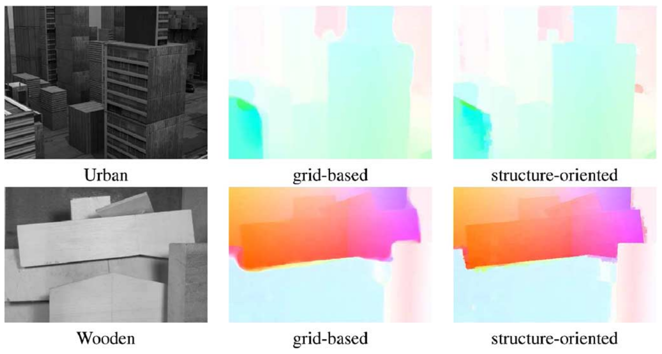

Fig. 5. Qualitative comparison of the grid-based and SO computation performance on sequences Urban and Wooden. Motion boundaries are evidently better preserved by the SO flow computation.

is 9541 s. Note that our optimization employs Jacobi iteration scheme, which is slower than the commonly adopted SOR.

We find that techniques such as data normalization [10] and image presmoothing [21] can further enhance the AAE and AEE accuracy of our baseline SO computation. To adapt image presmoothing to multiscale computation, we design the presmoothing filter f in level l according to the motion detail portion index P^{l+1}. Specifically, if $P^{l+1} < 0.01$, f is Gaussian with a standard deviation of 1, and if $0.01 \leq P^{l+1} < 0.02$, f is Gaussian with a standard deviation of 0.8; otherwise, f is an identity transform. Test results on the Middlebury training sequences (rows 5 and 6 of Table I) suggest that both data normalization (SO+DataNorm) and adaptive smoothing (SO+DataNorm+PreSmooth) benefit the SO flow computation on hidden texture sequences Venus, RubberWhale, Dimetrodon, Hydrangea, but not computer rendered ones such as Urban3, Grove3, and Grove2. This observation is also supported by the Middlebury evaluation (Table II and the evaluation website [34]). Despite varying results on individual sequences, the overall AAE and AEE ranks are lifted above the baseline computation and to a level competitive with comparable methods such as Classic++ and TV-L^1-improved. We therefore use the SO method with data normalization and presmoothing (SO+DataNorm+PreSmooth, row 6 of Table I) in practical applications. However, for the purpose of evaluating the oriented coordinate frame, the following discussions are based on our baseline SO cost functional and optimization scheme (SO, row 1 of Table I).

B. Why We Use Directional Convolution for Edge Detection (Directional Convolution Versus The Gradient)

As the SO coordinate frame is attached to local intrinsic directions, the detection of the edge direction and edge normal is fundamental to the flow estimation. Although there are prevalent edge detection methods such as using the gradient or the

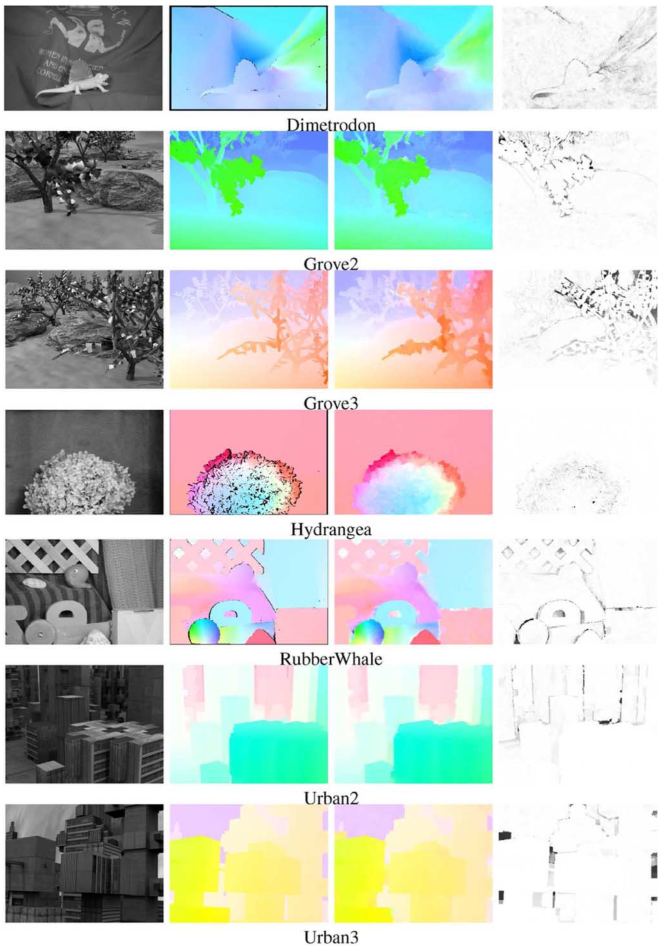

Fig. 6. Color coded results of the flow estimation by oriented flow computation on Middlebury training sequences. From left to right, the test images, the ground-truth flow, the computed flow, and the AAE images. In the grayscale error maps, darker intensity indicates larger errors.

local structure tensor's eigensystem, they derive the directional variation from the horizontal and vertical variations, which may be unreliable around directional boundaries. In Section III, we propose a new approach based on directional convolution. In this section, we compare the proposed PDC and RDC to the prevalent methods. To test the generality of these methods, we conduct this set of experiments on images that contain edges and lines in a variety of orientations. Fig. 7 shows the results on the test image Barbara, which consists of linelike textures of different directions and has been seeded with noise. Fig. 8 shows the results on the test image Lights, which consists of lines that densely cover the 2-D plane.

In these experiments, the structure tensor is defined by

$$\boldsymbol{ST} = \begin{bmatrix} \boldsymbol{G}_\sigma \otimes \left(E_x^2\right) & \boldsymbol{G}_\sigma \otimes \left(E_x E_y\right) \\ \boldsymbol{G}_\sigma \otimes \left(E_x E_y\right) & \boldsymbol{G}_\sigma \otimes \left(E_y^2\right) \end{bmatrix} \tag{35}$$

TABLE II
MIDDLEBURY EVALUATION OF THE SO COMPUTATION AND THE ENHANCED STRUCTURE ORIENTED COMPUTATION (SO+DATANORM+PRESMOOTH) AND COMPARABLE APPROACHES. BOLD NUMBERS INDICATE BEST PERFORMANCE. SEE [34] FOR DETAILS

AAE	Avg rank.	Army	Mequon	Schefflera	Wooden	Grove	Urban	Yosemite	Teddy
SO+DataNorm+PreSmooth	22.2	4.18	5.05	7.03	**3.15**	3.43	3.94	**2.14**	**3.71**
Classic++	23.3	3.37	3.28	**5.46**	3.63	**3.24**	4.65	3.09	4.64
Tv-L^1-improved	24.7	**3.36**	**2.82**	6.50	3.80	3.34	5.97	3.57	4.01
StructureOriented (SO)	25.3	4.54	4.73	7.77	3.56	3.46	**3.15**	2.61	4.39

Middlebury Evaluation of the AAE performance

AEE	Avg rank.	Army	Mequon	Schefflera	Wooden	Grove	Urban	Yosemite	Teddy
Classic++	20.8	0.09	0.23	**0.43**	0.20	**0.87**	**0.47**	0.17	0.79
SO+DataNorm+PreSmooth	23.4	0.11	0.34	0.55	**0.17**	0.96	1.34	**0.11**	**0.68**
Tv-L^1-improved	25.2	0.09	**0.20**	0.53	0.21	0.90	1.51	0.18	0.73
StructureOriented (SO)	25.5	0.12	0.33	0.61	0.18	0.93	0.98	0.12	0.73

Middlebury Evaluation of the AEE performance

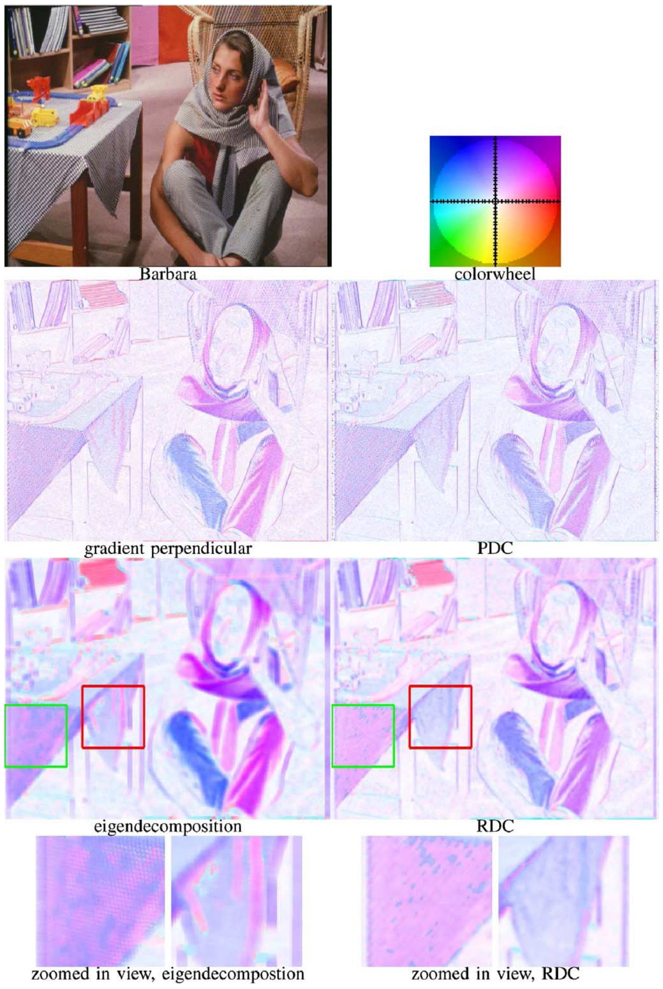

Fig. 7. Edge detection results by different methods on a test image Barbara. In this experiment, RDC shows its robustness to noise. Moreover, it outperforms the structure tensor eigendecomposition method in the table cloth area, where image intensity values change fast. Both methods use the same Gaussian convolution kernel.

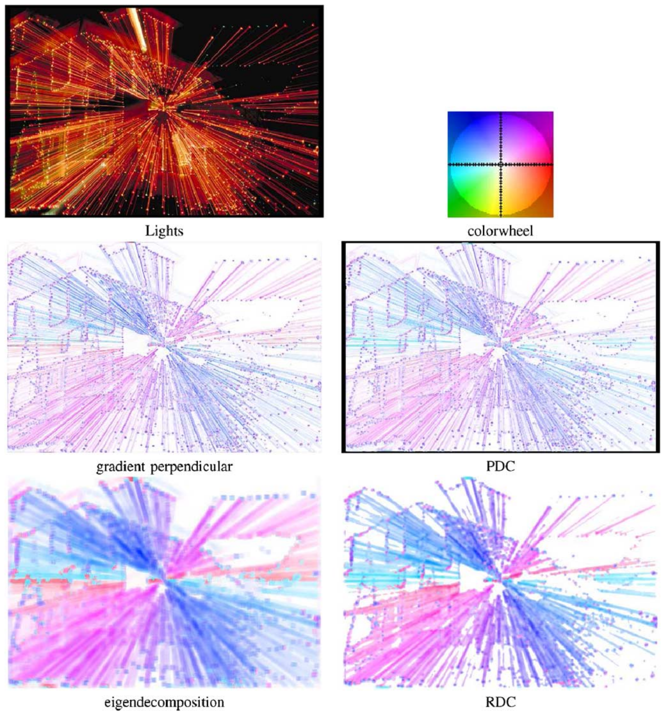

Fig. 8. Edge detection results by different methods on the test image Lights. It can be seen that RDC preserves the line boundaries more sharply than the structure tensor eigendecomposition, although both methods use the same Gaussian convolution kernel.

where $\boldsymbol{G}_\sigma$ is an 11×11 Gaussian convolution kernel and σ is given by $11/5 = 2.2$. The RDC method adopts the same $\boldsymbol{G}_\sigma$.

For clarity, we code the edge detection results by the Middlebury color wheel, with the edge directions represented by the hue values and the edge strength represented by the brightness. In the experiments, unit vectors $[\cos\theta, \sin\theta]$ and $[\cos(\theta + \pi), \sin(\theta + \pi)]$ are unified to $[\cos\theta, \sin\theta]$, $\theta \in [0, \pi)$ because both vectors lie on the same line. As the edge strength is a "relative" quantity, the computation of the edge strength varies from method to method. Specifically, the gradient method computes the edge strength as the gradient magnitude; the structure tensor eigendecomposition uses the square root of the largest eigenvalue; whereas PDC and RDC employ the intensity variation in the detected edge normal direction. For comparison convenience, we scale the edge strength to a fixed range of [0, 20].

The results on image Barbara show that PDC and the gradient method detect the edge directions unstably, particularly in the table cloth area. This demonstrates the sensitivity of the two methods to fast intensity variation and noise. Evidently, RDC and the structure tensor eigendecomposition method are more robust. Furthermore, a detailed comparison on the two example regions show that RDC performs more consistently in the presence of textures, as illustrated by Fig. 7.

On test image Lights, PDC performs comparably to the gradient method. This validates the adequacy of using 20 directions to quantize the orientation range $[0, \pi)$. As for their robust extensions, Fig. 8 shows that RDC preserves the edge boundaries more sharply than the structure tensor eigendecomposition.

C. Why We Compute Directional Derivatives Without Projection (Original Definition Versus Steerable Filtering)

The flow recovery quality is decided by the coefficient functions ω, ϖ, β_j, κ_j, γ_j, ξ_j, C_1, and C_2 in the numerical solution (33). These functions are expressed by the directional derivatives of intensity and motion. As proposed in Section III, our SO method computes the directional derivatives of a surface χ by

$$\frac{\partial \chi}{\partial \boldsymbol{e}} = \chi(Z + \boldsymbol{e}) - \chi(Z) \tag{36}$$

where $Z = [\zeta\ \eta]^T$. A prevalent estimation scheme for directional derivatives is to project the surface gradient to the direction $\boldsymbol{e}$, i.e.,

$$\frac{\partial \chi}{\partial \boldsymbol{e}} = \langle \nabla \chi, \boldsymbol{e} \rangle \tag{37}$$

which is a special case of steerable filtering [35].

To empirically compare the two schemes in terms of their contribution to the flow recovery accuracy, we compute the coefficient functions of (33) by (37), and recover the flow field with settings otherwise identical to our SO method. Row 4 of Table I (SO projection) presents the yielded AAE and AEE results on Middlebury training sequences. Compared with the results presented from the SO flow computation (Row 1 of Table I, SO), it can be seen that (36) leads to better or similar accuracy on all test sequences. We conduct this experiment under several different settings of λ values. Although the performance difference of the two schemes may vary on each sequence, the overall superior performance of scheme (36) is always observed.

Figs. 9 and 10 present the flow computed under steerable filtering scheme (37) on sequence HumanEva-II. Compared with this result, the flow computed under scheme (36) (see Fig. 3) preserves the motion boundary between the foot and leg more faithfully.

D. Nonlocal Regularization (Structure-Aligned Segmentation Versus Pairwise Weighting)

Section II-D proposes segmenting the nonlocal smoothing window into the inlier part and the outlier part by the intrinsic edge direction and imposing the nonlocal smoothness constraint only to the inlier part. Previous works generally suppress the outliers by pairwise weighting. It is interesting to compare the efficiency of the structure segmented and the weighted nonlocal regularization. However, as they are part of objective functionals that are formulated in different coordinate frames, it is hard to compare their performance without changing the other energy terms. We thus turn to the close connection between nonlocal smoothing and median filtering.

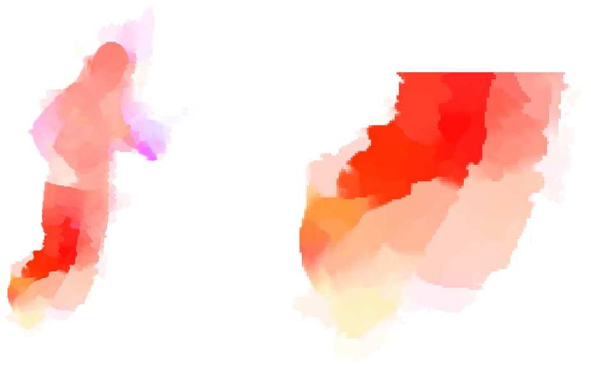

Fig. 9. The color coded flow obtained on Sequence HumanEva-II by the algorithm that employs steerable filtering (37) to compute directional derivatives. Compared with the result obtained by the algorithm that employs the original definition of directional derivative (36), scheme (37) results in underestimation of the foot's motion and blurry artifacts around the foot and leg. See Fig. 3 for comparison.

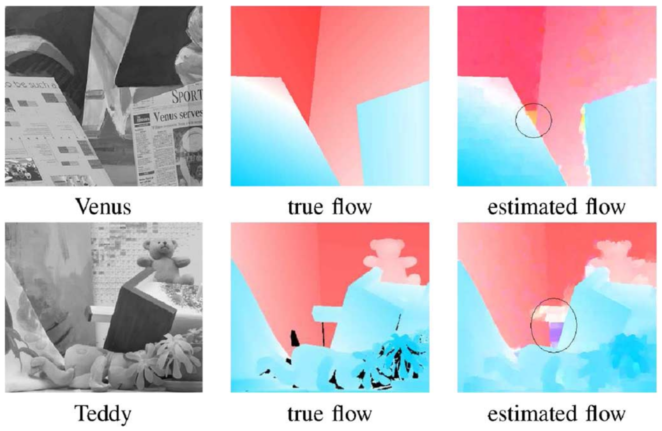

Fig. 10. Examples of motion junctions where SO computation may have inferior performance to the grid-based computation. Both sequences contain junction areas (circled), where three moving objects meet. In this situation, the detected dominant direction does not predict the motion boundary well, consequently the flow computation suffers errors in such areas.

In the case where ψ is the $l1$-norm penalizer, the nonlocal regularization is essentially equivalent to a median filtering operation on the intermediate flow obtained after each refining stage, although the nonlocal regularization leads to lower energy solutions than median filtering. The similar performance by the two schemes are analyzed by [14]. This means that the pairwise weighted nonlocal smoothing roughly corresponds to a weighted median filtering, and the structure-segmented nonlocal smoothing roughly corresponds to a median filtering in the inlier region W_{in}. Hence, we convert the comparison of the nonlocal regularization to the comparison of their corresponding median filtering. First, this allows the remaining experimental settings to remain the same. Second, this comparison provides future reference for applications that involve pairwise weighted median filtering.

We conduct the comparison on the platform of Sun's publicly accessible code [14], [36], particularly the "classic+nl-fast" function with the "pcg" solver, which computes the flow increment by a classic model and denoises the intermediate flow by a mixture of weighted median filtering and traditional median filtering. We replace the weighted median filtering with the

TABLE III
AAE AND AEE RESULTS OF CWF WITH DIFFERENT WINDOW SIZES AND CSF

	Grove3		RubberWhale		Dimetrodon		Hydrangea		Venus		Grove2		Urban2		Urban3	
	AAE	AEE	AAE	AEE	AAE	AEE	AAE	AEE	AAE	AEE	AAE	AEE	AAE	AEE	AAE	AEE
CSF	7.81	0.84	5.53	0.17	5.83	0.32	2.49	0.23	6.05	0.42	3.94	0.28	4.76	0.89	8.49	1.33
CWF-15	8.18	0.87	5.63	0.18	5.94	0.34	2.52	0.23	6.05	0.41	3.74	0.26	4.35	0.59	8.20	1.27
CWF-5	7.47	0.79	5.46	0.17	5.79	0.32	2.45	0.23	6.72	0.44	3.38	0.25	4.84	0.79	9.17	1.26

structure-segmented median filtering and compare their effects on the flow computation accuracy and their running time. In the following discussion, we refer to the programs with the two filtering schemes as classic+weighted-fast (CWF) and classic+segmented-fast (CSF).

In this set of experiments, the weight function of the CWF program is defined by the intensity contrast. Specifically

$$\omega_k = \exp\left(-\frac{(E_k - E_o)^2}{2\sigma^2}\right) \tag{38}$$

where E_o denotes the intensity of the current pixel and E_k denotes the kth pixel in the nonlocal window. $\sigma = 5$, as in [36]. The original code of [36] employs a more sophisticated weight function, which is defined by the spatial distance, intensity contrast (in the Lab space, if RGB channels are available), and occlusion measurement. This function leads to superior performance but at the price of a higher computation cost. The neighborhood size of the structure-segmented filtering is set to 7 × 7, which means that the inlier region W_{in} has 24 pixels. The rest of the code of both programs remain identical to the original, and all parameters take the default values. To record the time consumed, we set on the profile in Matlab.

Table III reports their AAE and AEE results on eight Middlebury training sequences. As the weighted median filtering uses all the pixels in the nonlocal window (15 × 15), whereas the structure-segmented median filtering uses the window partially (24 pixels), it was expected that the structure-segmented median filtering trades accuracy for computation time. However, compared with CWF, CSF has superior accuracy on sequence Grove3; similar accuracy on sequences RubberWhale, Hydrangea, Dimetrodon, and Venus; and inferior accuracy on sequences Grove2, Urban2, and Urban3. The performance difference on all sequences is smaller than 0.4° in AAE, which is moderate. On sequence Urban, each call of the weighted median filtering takes 6.46 s on average; whereas each call of the structure-segmented median filtering takes about 1.72 s. To summarize, out of the eight training sequences, the structure-segmented median filtering leads to moderately inferior performance on three sequences, and comparable performance on the rest; but is about 3.75 times faster than the intensity-weighted median filtering.

To compare the numerical performance of CWF and CSF at similar computation cost, we also run a 5 × 5 weighted median filtering (i.e., CWF-5) by changing the window size parameter in to the code in [36] so that both schemes use about the same number of pixels. Each call of CWF-5 takes 2.61 s on the average. This is about 2.48 times faster than one call of a 15 × 15 weighted median filtering but is still 1.52 times slower than the proposed scheme. Compared with CWF-15, CWF-5 significantly deteriorates the computation accuracy on Urban2, Urban3, and Venus.

VI. CONCLUSION

In this paper, we have proposed that optical flow should be calculated at each pixel in an adaptive coordinate system, which is aligned with the least local curvature direction. We have shown that in this SO coordinate system, data constraints based on the assumption of gradient constancy become robust to fast rotation. In addition, local smoothness constraints preserve motion boundaries with greater fidelity, and nonlocal smoothness constraints can be imposed with more computational efficiency.

We have also proposed new approaches to computing directional derivatives and inferring coordinate axes at each pixel and have shown that these lead to more accurate recovery of flow, compared with projection based methods.

The use of a locally oriented coordinate frame can be incorporated into many existing flow computation methods without major alteration to the algorithm structure. We anticipate that this will also lead to further improvement of results, although we leave the implementation to future work.

REFERENCES

[1] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. DARPA Image Understand. Workshop*, 1981, pp. 121–130.

[2] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Proc. Eur. Conf. Comput. Vis.*, 2004, pp. 25–36.

[3] P. Laskov and C. Kambhamettu, "Curvature-based algorithms for nonrigid motion and correspondence estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 10, pp. 1349–1354, Oct. 2003.

[4] L. Pizarro, P. Mrázek, S. Didas, S. Grewenig, and J. Weickert, "Generilised nonlocal image smoothing," *Int. J. Comput. Vis.*, vol. 90, no. 1, pp. 62–87, Oct. 2010.

[5] M. Tistarelli and G. Sandini, "On the advantage of polar and log-polar mapping for direct estimation of time-to-impact from optical flow," *Pattern Anal. Mach. Intell.*, vol. 15, no. 4, pp. 401–410, Apr. 1993.

[6] K. Daniilidis and V. Kruger, "Optical flow computation in the log-polar plane," in *Comput. Anal. Images Patterns*, 1995, pp. 65–72.

[7] H. T. Ho and R. Goecke, "Optical flow estimation using Fourier Mellin transform," in *Proc. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–8.

[8] H. Nagel and W. Enkelmann, "An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 5, pp. 565–593, Sep. 1986.

[9] D. Sun, S. Roth, J. Lewis, and M. Black, "Learning optical flow," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 83–97.

[10] H. Zimmer, A. Bruhn, and J. Weickert, "Optic flow in harmony," *Int. J. Comput. Vis.*, vol. 93, no. 3, pp. 368–388, Jul. 2011.

[11] K. Lee, D. Kwon, I. Yun, and S. Lee, "Optical flow estimation with adaptive convolution kernel prior on discrete framework," in *Proc. Comput. Vis. Pattern Recognit.*, 2010, pp. 2504–2511.

[12] M. Werlberger, T. Pock, and H. Bischof, "Motion estimation with non-local total variation regularization," in *Proc. Comput. Vis. Pattern Recognit.*, 2010, pp. 2464–2471.

[13] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi, "Bilateral filtering-based optical flow estimation with occlusion detection," in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 211–224.

[14] D. Sun, S. Roth, and M. Black, "Secrets of optical flow estimation and their principles," in *Proc. Comput. Vis. Pattern Recognit.*, 2010, pp. 2432–2439.

[15] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artif. Intell.*, vol. 17, pp. 185–203, 1981.
[16] A. Wedel, T. Pock, J. Braun, U. Franke, and D. Cremers, "Duality TV-L1 flow with fundamental matrix prior," in *Proc. Image Vis. Comput. New Zealand*, 2008, pp. 1–6.
[17] H. Zimmer, A. Bruhn, J. Weickert, L. Valgaerts, A. Salgado, B. Rosenhahn, and H.-P. Seidel, "Complementary optic flow," in *Proc. Energy Minimization Methods Comput. Vis. Pattern Recognit.*, 2009, pp. 207–220.
[18] M. Black and P. Anandan, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow-fields," *Comput. Vis. Image Understand.*, vol. 63, no. 1, pp. 75–104, Jan. 1996.
[19] D. Shulman and J. Herve, "Regularization of discontinuous flow fields," in *Proc. Workshop Vis. Motion*, 1989, pp. 81–86.
[20] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof, "Anisotropic Huber-L1 optical flow," in *Proc. Brit. Mach. Vis. Conf.*, 2009.
[21] J. Barron, D. J. Fleet, and S. Beauchemin, "Performance of optical flow techniques," *Int. J. Comput. Vis.*, vol. 12, no. 1, pp. 43–77, Feb. 1994.
[22] J. Weickert, A. Bruhn, T. Brox, and N. Papenberg, "A survey on variational optic flow methods for small displacements," *Math. Models Registration Appl. Med. Imag.*, vol. 10, pp. 103–136, 2006.
[23] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.
[24] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. Freeman, "SIFT flow: Dense correspondence across different scenes," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. III: 28–III: 42.
[25] X. Ju, M. J. Black, and A. D. Jepson, "Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency," in *Proc. Comput. Vis. Pattern Recognit.*, 1996, pp. 307–314.
[26] T. Nir, A. Bruckstein, and R. Kimmel, "Over-parameterized variational optical flow," *Int. J. Comput. Vis.*, vol. 76, no. 2, pp. 205–216, Feb. 2008.
[27] W. Trobin, T. Pock, D. Cremers, and H. Bischof, "An unbiased second-order prior for high-accuracy motion estimation," in *Proc. 30th German Symp. Pattern Recognit.*, 2008, pp. 396–405.
[28] Å. Björck, *Numerical Methods for Least Squares Problems*. Philadelphia, PA: SIAM, 1996.
[29] S. Baker, S. Roth, D. Scharstein, M. Black, J. Lewis, and R. Szeliski, "A database and evaluation methodology for optical flow," *Int. J. Comput. Vis.*, vol. 92, no. 1, pp. 1–31, Mar. 2011.
[30] L. Sigal, A. Balan, and M. Black, "Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion," *Int. J. Comput. Vis.*, vol. 87, no. 1/2, pp. 4–27, Mar. 2010.
[31] T. Brox, C. Bregler, and J. Malik, "Large displacement optical flow," in *Proc. Comput. Vis. Pattern Recognit.*, 2009, pp. 41–48.
[32] F. Steinbruecker, T. Pock, and D. Cremers, "Large displacement optical flow computation without warping," in *Proc. Int. Conf. Comput. Vis.*, 2009, pp. 1609–1614.
[33] L. Xu, J. Jia, and Y. Matsushita, "Motion detail preserving optical flow estimation," in *Proc. Comput. Vis. Pattern Recognit.*, 2010, pp. 1293–1300.
[34] Middlebury evaluation [Online]. Available: http://vision.middlebury.edu/flow/eval/
[35] W. T. Freeman and E. H. Adelson, "The design and use of steerable filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 9, pp. 891–906, Sep. 1991.
[36] Publicly available code of optical flow computation methods [Online]. Available: http://www.cs.brown.edu/dqsun

Yan Niu received the Ph.D. degree from the University of Adelaide, Adelaide, Australia, in 2010, where she held the Australia IPRS scholarship.

She is currently a Lecturer with the College of Computer Science and Technology, Jilin University, Changchun, China. Her research interests include inverse problems in computer vision, image correspondence, and the application of numerical partial differential equations.

Anthony Dick received the Ph.D. degree from the University of Cambridge, Cambridge, U.K., in 2002, where he worked on problems in 3-D reconstruction of architecture from images.

He is currently a Senior Lecturer with the University of Adelaide, Adelaide, Australia. His research interests include image-based modeling, automated video surveillance, image search, and the intersection of computer vision and graphics.

Mike Brooks received the Ph.D. degree in computer vision from the University of Essex, Colchester, U.K., in 1983, and has since worked on a range of problems including shape from shading, stereo, structure from motion, and visual surveillance.

He is currently a Professor of Computer Science and the Deputy Vice Chancellor for Research with the University of Adelaide, Adelaide, Australia.

Dr. Brooks is a Fellow of the Australian Academy of Technological Sciences and Engineering and the Australian Computer Society and serves on the Board of National ICT Australia.