# Compass Rose: A Rotational Robust Signature for Optical Flow Computation

Yan Niu, Anthony Dick, and Michael Brooks

***Abstract*—This paper proposes a new image signature, called Compass Rose, which is particularly suited to optical flow computation. It is differentiable, fast to compute, and robust to additive illumination changes, translation, and fast rotation. We design a sparse flow computation system based on the invariance of the Compass Rose signatures. This is then extended to dense motion estimation by the addition of an optional diffusion step. Quantitative testing on several benchmark sequences shows the Compass Rose attains higher accuracy than the traditional flow signatures under a range of conditions. Finally, we demonstrate its application to human motion estimation, which is challenging for optical flow methods due to fast limb rotation.**

***Index Terms*—Data constraint, image signature, motion estimation, optical flow, rotational robustness.**

## I. Introduction

OPTICAL FLOW is an important technique to register a series of gradually changing images. The computation of optical flow is typically formulated as finding the optimal motion that meets a set of constraints with the least error [1]. Any flow computation functional should have at least one data constraint, which infers the optical flow (i.e., the perceived motion) of an image point by assuming that a pair of corresponding points should have a common signature value. The most frequently used signature is pixel intensity, but this fails to remain constant when the lighting condition changes. To compensate for this sensitivity, some data constraints use the gradient vector [2], which is unaffected by additive illumination changes but varies under rotation [3]. The rotational and scale invariant feature descriptor SIFT [4] is integrated with optical flow computation in [5] for scene matching. Recently, Tola–Lepetit–Fua [6] proposed a feature descriptor named DAISY for wide baseline stereo matching. The DAISY descriptor replaces SIFT's histogram of gradients by the dense and quick Gaussian convolution. It is also robust to rotation, as it uses the partial derivatives in eight intrinsic directions,[1] which in [6] consist of the local edge direction and the directions that are offset from the local edge direction by $[45°, 90°, \cdots, 315°]$ degrees, respectively. The directional derivatives in DAISY are obtained by projecting the gradient vector to these intrinsic directions. Such a projection implicitly assumes that the brightness function is continuously differentiable [8], which causes redundancy in the feature descriptor. Moreover, as SIFT and DAISY are not differentiable, they are not applicable for accurate optical flow computation.

In this paper, we propose a differentiable and rotationally robust signature, which is also composed of partial derivatives in the eight intrinsic directions similar to DAISY. However, different from the previous works, these directional derivatives are computed from the image values that are immediately available on the image lattice, without interpolation (such as in our earlier work [8]) or projection (such as in DAISY). Our computation scheme is based on the observation that an $11 \times 11$ patch contains 40 integer vectors (Fig. 1), which cover the directions in the 2-D plane densely and quite evenly and can be grouped into five-oriented eight-point compass roses (Fig. 2). Therefore, the eight intrinsic directions of a local patch can always be quantized by one of these compass roses with a negligible error. The advantages of using these integer vectors are that: first, the corresponding directional derivatives are easy to compute; second, directional derivatives computed in such a way that they capture the image structure more faithfully than using interpolation [8] or projection [6]. This benefits motion boundary preservation.

The invariance of the proposed signature provides a data constraint for optical flow computation. In this paper, we integrate it with a local affine motion model to construct an over-determined linear system and recover the flow vector by the optimal solution. This formulation recovers a flow vector independent of flow vectors at other pixels (although it does depend on neighboring pixels' intensity). Thus, it enables tracking a set of feature points sparsely. The proposed data constraint (and even the linear system) can also be integrated with smoothness constraints to exploit the spatial correlation of the motion field. Such a formulation recovers all flow vectors jointly, because each flow vector is constrained by its neighboring flow vectors. The joint flow

Manuscript received January 16, 2013; revised April 24, 2013 and June 18, 2013; accepted July 9, 2013. Date of publication August 6, 2013; date of current version January 3, 2014. This work was supported in part by the National Natural Science Foundation of China under Grant NSFC61103090 and in part by the Australian Research Council under Grant DP1094764. This paper was recommended by Associate Editor R. C. Lancini.

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. Brooks are with School of Computer Science, the University of Adelaide, Adelaide, SA 5005, Australia (e-mail: anthony.dick@adelaide.edu.au; michael.brooks@adelaide.edu.au).

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/TCSVT.2013.2276854

[1]Intrinsic direction is a concept in intrinsic geometry. We refer the readers to [7] for a precise definition.

recovery generally achieves higher average accuracy than its independent recovery counterpart [9]. However, joint flow computation is associated with heavier computational load. In many cutting-edge applications [10]–[13], independent sparse feature tracking[2] is still preferred to joint flow computation, as the time budget for this intermediate step may be limited and a sparse computation is probably sufficient. Therefore, we focus on the independent flow recovery system and solve it by a conventional routine [17]; on top of that, we suggest methods to handle motion boundaries and joint flow recovery.

We present the Compass Rose signature in Section II. Section III applies this signature to optical flow computation. Section IV evaluates the flow computation performance on independent feature tracking, joint motion estimation, and human motion estimation. The paper is concluded by Section V.

## II. Compass Rose: Rotational Invariant Signature

Let $E(x, y) : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ be the intensity function of an image. It is common to compute the image variation at a pixel $(i, j) \in \mathbb{Z} \times \mathbb{Z}$ in the horizontal and vertical direction by $[E(i+1, j) - E(i, j)]$ and $[E(i, j+1) - E(i, j)]$, respectively (with $i$ indexing the $i$th column and $j$ indexing the $j$th row of the image grid). To compute the variation in the diagonal direction, a seemingly plausible way is to project $[E(i+1, j) - E(i, j), E(i, j+1) - E(i, j)]$ to the direction $\frac{1}{\sqrt{2}}[1, 1]$. By doing this, one implicitly assumes that the image intensity is sufficiently smooth, which is not always true in practice. It is better to use $[E(i+1, j+1) - E(i, j)]$, as it captures image structure more faithfully and the involved intensity values are directly available on the image lattice. Obviously, within a local $3 \times 3$ patch, we can compute the variations in eight directions (Fig. 2a) in such a way. Enlarging the local patch to $7 \times 7$, this computation can be applied to 24 directions (Fig. 2a–c). For example, image variation in the direction of $63.44^\circ$ can be obtained by $\frac{1}{\sqrt{5}}[E(i+1, j+2) - E(i, j)]$ rather than projecting the gradient to $[\cos\frac{63.44\pi}{180}, \sin\frac{63.44\pi}{180}] = \frac{1}{\sqrt{5}}[1, 2]$.

Our feature signature is defined on the $11 \times 11$ patch, for the particular reason that the 40 integer vectors contained in the patch cover the directions in the 2-D plane densely and quite evenly, as shown in Fig. 1. Specifically, the 10 vectors that cover the first quadrant[3] are

$$\boldsymbol{e}_0 = [1, 0];\ \boldsymbol{e}_1 = [5, 1];\ \boldsymbol{e}_2 = [3, 1];\ \boldsymbol{e}_3 = [2, 1];\ \boldsymbol{e}_4 = [3, 2]$$
$$\boldsymbol{e}_5 = [1, 1];\ \boldsymbol{e}_6 = [2, 3];\ \boldsymbol{e}_7 = [1, 2];\ \boldsymbol{e}_8 = [1, 3];\ \boldsymbol{e}_9 = [1, 5].$$

Their orthogonal vectors cover the second quadrant, as shown in Fig. 1.

Among these vectors, the largest angle between two neighboring vectors is $11.31^\circ$ (e.g., between $\boldsymbol{e}_0$ and $\boldsymbol{e}_1$) and the smallest is $7.13^\circ$ (e.g., between $\boldsymbol{e}_3$ and $\boldsymbol{e}_4$). Thus, the quantization distortion is smaller than $5.7^\circ = 11.31^\circ/2$. This is comparable to using the 40 evenly distributed quantization directions in [8], where the quantization distortion upper limit is $4.5^\circ$.

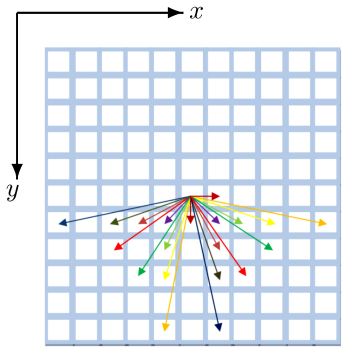



Fig. 1. Twenty vectors that fall on the integer positions of the image grid, covering the first and second quadrant. Pair of vectors that are of the same color are orthogonal.

Together with the opposite vectors, the quantization vectors in Fig. 1 can be grouped into five sets, as shown in Fig. 2. Each set contains four cardinal and four ordinal directions of an oriented compass rose. Therefore, no matter how the local patch is rotated, the eight intrinsic directions can be approximated with negligible error ($\leq 5.7^\circ$) by one of the five compass roses. In other words, patch size larger than $11 \times 11$ is unnecessary in most cases.

In each of these quantization directions $\boldsymbol{d}$, the directional derivative computation of the image brightness function $E(x, y)$ is defined by

$$E_{\boldsymbol{d}} = \frac{\partial E}{\partial \boldsymbol{d}} \approx \frac{E(X + \boldsymbol{d}) - E(X)}{\|\boldsymbol{d}\|} \tag{1}$$

where $X = [x, y]$. With $\boldsymbol{d}$ represented by integers, pixel $X + \boldsymbol{d}$ falls on the image grid. Thus, (1) utilizes intensity values that are immediately available on the image lattice. For example, the first-order derivative in the direction of $\boldsymbol{e}_2$ can be obtained by simply filtering the image with the high-pass filter

$$K_{\frac{\partial}{\partial \boldsymbol{e}_2}} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

followed by a uniform scaling of $1/\sqrt{5}$. Compared to the differencing scheme in [8], which needs bilinear interpolation to approximate the intensity values at subpixels, (1) avoids the blurring of sharp edges that results from the smooth interpolation. However, as $E_{\boldsymbol{d}}$ is computed by the image values immediately available at the image grid, the image resolution affects the final differencing values. If the image is obtained with high-sampling rate, the right-hand side of (1) is a more accurate approximation of $E_{\boldsymbol{d}}$. Conversely, sample aliasing in the image may lead to biased approximation. Therefore, this scheme suits applications where the image resolution meets the Nyquist sampling criterion [18]. That is, the sampling frequency is no lower than twice the highest signal frequency.[4]

If this requirement is satisfied, the local edge normal direction is detected by the eigen-decomposition of the structure

[2]Feature tracking may have slightly different meaning in different context. Here, it is meant in the same vein as Kanade–Lucas–Tomasi tracker [14], which extracts interesting points from the initial frame and estimates their positions in the subsequent frames by optical flow (see [15, ch. 4] and [16, p. 6]).

[3]The coordinate frame in this paper has its positive $x$ direction pointing to the right, and $y$ direction pointing down.

[4]Having said that, in our pyramidal optical flow computation, where the images are subsampled to lower resolutions, the scale difference does not pose a problem.

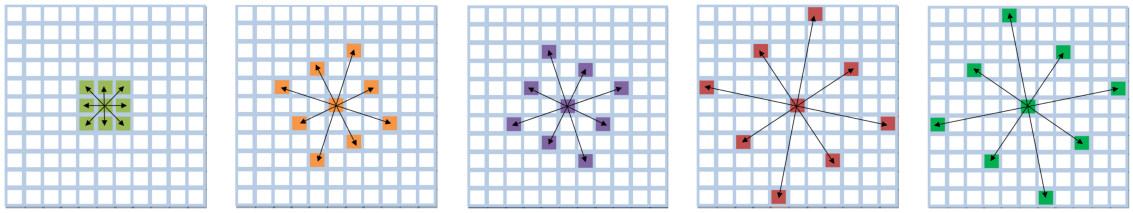



Fig. 2. Quantization vectors can be grouped into five sets. Each set contains four cardinal and four ordinal directions of a rotated compass rose.

tensor constructed on the local $N \times N$ surrounding window $\mathcal{W}$ (see [19, p. 82] for details)

$$\begin{bmatrix} \sum_{\mathcal{W}} E_x^2 & \sum_{\mathcal{W}} E_x E_y \\ \sum_{\mathcal{W}} E_x E_y & \sum_{\mathcal{W}} E_y^2 \end{bmatrix}$$

and modulated to the range of $[0, \pi]$. Next, the pixel is assigned to one of the oriented compass roses in Fig. 2 that the normal direction belongs to.

We now extend the image $E(x, y)$ to a video sequence $E(x, y, t)$, with $t$ symbolizing the time dimension. Let $\boldsymbol{n}(t)$ be the quantized normal direction of $E$ at position $(x, y, t)$. Denote the 8 directions of its associated compass rose, in clockwise order, by $\boldsymbol{d}_0(t), \boldsymbol{d}_1(t), \cdots, \boldsymbol{d}_7(t)$, with $\boldsymbol{d}_0(t) = \boldsymbol{n}(t)$. The first order partial derivatives in these directions form a vector $\boldsymbol{f}(x, y, t)$:

$$\boldsymbol{f}(x, y, t) = \left[ \frac{\partial E(x, y, t)}{\partial \boldsymbol{d}_0(t)}, \cdots, \frac{\partial E(x, y, t)}{\partial \boldsymbol{d}_7(t)} \right]. \tag{2}$$

Intuitively, this signature can be viewed as always aligning the north direction of an 8-point compass rose with the local normal direction,[5] and computing the directional intensity variation using the 8-points. For this resemblance, we name signature vector $\boldsymbol{f}$ Compass Rose.

## III. Flow Computation Based on the Compass Rose Signature

### A. Independent Flow Computation System

The Compass Rose signature $\boldsymbol{f}$ defined in (2) remains constant under additive illumination changes, translation, and rotation. This invariance connects a point $(x, y, t)$ with its corresponding point $(x + u, y + v, t + 1)$ in the next frame by

$$\boldsymbol{f}(x, y, t) = \boldsymbol{f}(x + u, y + v, t + 1). \tag{3}$$

Assuming $u$ and $v$ are small (large displacement is addressed by pyramidal flow refinement [17]), the Taylor expansion of $\boldsymbol{f}(x + u, y + v, t + 1)$ leads to

$$\boldsymbol{f}_x u + \boldsymbol{f}_y v + \boldsymbol{f}_t = 0 \tag{4}$$

where the subscripts denote the corresponding partial derivatives. Equation (4) imposes a linear constraint on the optical flow vector at pixel $(x, y, t)$. One can integrate this data constraint with smoothness constraints to compute the flow field jointly and smoothly, in the way that Horn–Schunck and most contemporary flow computation schemes do. However, the objective functional formulated in such a way has to be optimized over the whole image, since the computation of one flow vector relies on the computation of neighboring flow vectors. This is more time-consuming than computing the flow vectors of a sparse set of features. In fact, many cutting-edge applications employ independent feature tracking to achieve the trade-off between computation time and information sufficiency. Aiming at such applications, we construct a linear system that consists purely of data constraints based on the Compass Rose signature, without employing a smoothness constraint. In Section III-D, we provide an optional step that addresses the aperture problem (i.e., the system is rank deficient due to the lack of local textures [19]), whereby the independent feature tracking can be extended to joint flow recovery, without altering our flow computation structure.

Previous independent feature tracking schemes generally assume a locally constant translation model [2], [17], [14]. In this paper, as we aim at recovering fast 2-D rotation (significant orientation change of the apparent image pattern between consecutive frames), we adopt the more general locally affine motion model, which was discussed in a few previous works [20], [21] related to image motion and was applied to monitoring feature loss in [14]. We express the flow vector $(u, v)$ at $(x, y)$ by

$$\begin{aligned} u &= \alpha_1 x + \alpha_2 y + \alpha_3 \\ v &= \alpha_4 x + \alpha_5 y + \alpha_6 \end{aligned} \tag{5}$$

where $\alpha_1, \cdots, \alpha_6$ represent the affine motion parameters. Let $(x^{(k)}, y^{(k)})$ denote the $k$th neighbor in the local surrounding $N \times N$ patch. If $(x^{(k)}, y^{(k)})$ and $(x, y)$ correspond to the same moving object, flow vector $(u^{(k)}, v^{(k)})$ at $(x^{(k)}, y^{(k)})$ satisfies

$$\begin{aligned} u^{(k)} &= \alpha_1 x^{(k)} + \alpha_2 y^{(k)} + \alpha_3 \\ v^{(k)} &= \alpha_4 x^{(k)} + \alpha_5 y^{(k)} + \alpha_6. \end{aligned} \tag{6}$$

Equations (5) and (6) lead to

$$\begin{aligned} u^{(k)} &= u + \alpha_1 \Delta x^{(k)} + \alpha_2 \Delta y^{(k)} \\ v^{(k)} &= v + \alpha_4 \Delta x^{(k)} + \alpha_5 \Delta y^{(k)} \end{aligned} \tag{7}$$

where $\Delta x^{(k)} = x^{(k)} - x$ and $\Delta y^{(k)} = y^{(k)} - y$.

Since the Compass Rose descriptor $\boldsymbol{f}^{(k)}$ at $(x^{(k)}, y^{(k)})$ is also invariant, $(u^{(k)}, v^{(k)})$ meets

$$\boldsymbol{f}_x^{(k)} u^{(k)} + \boldsymbol{f}_y^{(k)} v^{(k)} + \boldsymbol{f}_t^{(k)} = 0. \tag{8}$$

[5] Theoretically, one can also align the north direction of the compass rose with the local isophote direction. But, in practice, it is more robust to align it with the normal direction.

Substituting (7) into (8), we have

$$\begin{aligned} -\boldsymbol{f}_t^{(k)} &= \boldsymbol{f}_x^{(k)}(u+\alpha_1\Delta x^{(k)}+\alpha_2\Delta y^{(k)})+ \\ &+ \boldsymbol{f}_y^{(k)}(v+\alpha_4\Delta x^{(k)}+\alpha_5\Delta y^{(k)}). \end{aligned} \tag{9}$$

If $(x^{(k)}, y^{(k)})$ and $(x, y)$ have different motion patterns, $(u^{(k)}, v^{(k)})$ does not satisfy (6). Consequently, (9) provides eight invalid constraints, which may severely distort the recovery of $(u, v)$. To suppress such outliers caused by the motion boundaries, which often coincide with intensity boundaries, the equations are weighted by intensity consistency. That is,

$$\begin{aligned} -\omega^{(k)}\boldsymbol{f}_t^{(k)} &= \omega^{(k)}\boldsymbol{f}_x^{(k)}(u+\alpha_1\Delta x^{(k)}+\alpha_2\Delta y^{(k)})+ \\ &+ \omega^{(k)}\boldsymbol{f}_y^{(k)}(v+\alpha_4\Delta x^{(k)}+\alpha_5\Delta y^{(k)}) \end{aligned} \tag{10}$$

with

$$\omega^{(k)} = \exp^{\frac{-\left|E(x+\Delta x^{(k)},y+\Delta y^{(k)},t)-E(x,y,t)\right|}{\sigma}}. \tag{11}$$

The parameter $\sigma$ controls the softness of the weighting effect: a large $\sigma$ treats most neighbors as inliers, whereas a small $\sigma$ treats most neighbors as outliers. Our setting of $\sigma$ is based on the observation that in 256 gray level images, points that have color contrast smaller than eight levels are very likely to come from the same object (a similar criterion is applied to mean-shift image segmentation [22]). Conversely, higher contrast is more likely to be associated with a motion boundary. Empirically, varying $\sigma$ from 10 to 20 yields very slight difference to the flow recovery results. Here, we set $\sigma = 16$.

Equation (10) means that each neighboring pixel contributes eight constraints for the motion parameters $\boldsymbol{X} = [u, v, \alpha_1, \alpha_2, \alpha_4, \alpha_5]^T$. Thus, the constraints provided by all neighboring pixels form an over-determined system $A\boldsymbol{X} = \boldsymbol{b}$, where $A \in \mathbb{R}^{8N^2\times 6}$ and $\boldsymbol{b} \in \mathbb{R}^{8N^2}$. Particularly, the $(8k+1)$th to $8(k+1)$th $(k = 0, \cdots, N^2-1)$ rows of $A$ are

$$\omega^{(k)}\left[\boldsymbol{f}_x^{(k)} \quad \boldsymbol{f}_y^{(k)} \quad \boldsymbol{f}_x^{(k)}\Delta x^{(k)} \quad \boldsymbol{f}_x^{(k)}\Delta y^{(k)} \quad \boldsymbol{f}_y^{(k)}\Delta x^{(k)} \quad \boldsymbol{f}_y^{(k)}\Delta y^{(k)}\right]$$

and the $(8k+1)$th to $8(k+1)$th entries of $\boldsymbol{b}$ are $-\omega^{(k)}\boldsymbol{f}_t^{(k)}$.

### B. Pyramidal Implementation Details

As is standard for flow computation algorithms, we adopt the pyramidal coarse-to-fine flow refinement strategy [17] to tackle large displacement. Briefly, a Gaussian pyramid is constructed from the original image pair, with a subsample factor of 2, until the short dimension of the subsampled images has about 30 pixels [23]. To prevent sample aliasing, a $7\times 7$ Gaussian smoothing filter with standard deviation of 1.2 is applied before each subsampling operation. The flow computation, being initialized by zeros, starts from the coarsest level $L$ and is refined for up to $S$ stages in each level. In our implementation, we set $S = 5$. In this context, the objective functional is expressed by

$$A^{l,s+1}\delta\boldsymbol{X}^{l,s+1} = \boldsymbol{b}^{l,s+1} \tag{12}$$

where $l$ and $s$ index the level and stage; $A^{l,s+1}$ and $\boldsymbol{b}^{l,s+1}$ are obtained by warping the $N\times N$ surrounding patch centered at point $(x+u^{l,s}, y+v^{l,s}, t+1)$ in the second image; and

$$\delta\boldsymbol{X}^{l,s+1} = \left[\delta u^{l,s+1}, \delta v^{l,s+1}, \delta\alpha_1^{l,s+1}, \delta\alpha_2^{l,s+1}, \delta\alpha_4^{l,s+1}, \delta\alpha_5^{l,s+1}\right]^T$$

with $\delta$ symbolizing the increment in the corresponding motion parameter. In our implementation, the local neighborhood has size $7\times 7$ on the top level and is increased by 2 pixels in each dimension in each lower level. Within the level $l$, the flow vector is updated by

$$\begin{bmatrix} u^{l,s+1} \\ v^{l,s+1} \end{bmatrix} = \begin{bmatrix} u^{l,s} \\ v^{l,s} \end{bmatrix} + \begin{bmatrix} \delta u^{l,s+1} \\ \delta v^{l,s+1} \end{bmatrix} \tag{13}$$

and at the end of level $l$, the flow vector is propagated to level $l-1$ by

$$\begin{bmatrix} u^{l-1,0}(x,y) \\ v^{l-1,0}(x,y) \end{bmatrix} = 2\begin{bmatrix} u^{l,S}(\left[\frac{x}{2}\right], \left[\frac{y}{2}\right]) \\ v^{l,S}(\left[\frac{x}{2}\right], \left[\frac{y}{2}\right]) \end{bmatrix}. \tag{14}$$

### C. Adaptive Solution Based on Motion Inconsistency

An independent flow computation system formulated by $A\boldsymbol{X} = \boldsymbol{b}$ is conventionally solved by its least squares error (LSE) solution $\boldsymbol{X}^o$, which minimizes $\|A\boldsymbol{X}-\boldsymbol{b}\|_2$. However, the LSE solution is sensitive to outliers, which contribute invalid constraints for $\boldsymbol{X}$. Mathematically, in this case $A\boldsymbol{X} = \boldsymbol{b}$ does not have a solution and is termed as inconsistent in linear algebra (see [24, p. 277]). This can be intuitively viewed as the inliers and outliers providing inconsistent constraints. The LSE of an inconsistent system is a severely erroneous estimation of the true motion parameters. A robust alternative is to seek the least absolute error (LAE) solution, which minimizes $\|A\boldsymbol{X}-\boldsymbol{b}\|_1$. Yet handling the nondifferentiability of the $\ell_1$ penalizer involves extra computational cost. A trade-off between the cost and robustness is to use the LSE solution in general and the LAE solution only if system inconsistency is detected. Theoretically, the system is inconsistent if and only if the rank of matrix $[A|\boldsymbol{b}]$ is higher than the rank of $A$ [24]. Note that $A\boldsymbol{X}^o$ is the projection of $\boldsymbol{b}$ onto the column space of $A$ [25, p. 106]. Thus, $m = (\|A\boldsymbol{X}^o-\boldsymbol{b}\|_2 / \|\boldsymbol{b}\|_2) \in [0,1]$ continuously measures the component of $\boldsymbol{b}$ that falls into the null space of $A$. For extreme instances, $m = 0$ indicates that $\boldsymbol{b}$ is in the column space of $A$, and hence $[A|\boldsymbol{b}]$ and $A$ have the same rank, i.e., the system is consistent. Conversely, $m = 1$ indicates that the system is inconsistent. Thus, we use $m$ to measure the system inconsistency.[6] More specifically, we propose computing $m^{l,s+1}$ based on the LSE solution of (12); if $m^{l,s+1}$ is larger than a predefined threshold $T$ (its definition is discussed in Section IV), we proceed to compute the LAE solution by iteratively reweighted least squares (IRLS); otherwise we adopt the LSE solution. In our implementation, the weight function for IRLS takes the form of $\exp^{-|r^{(j)}|}$, where $r^{(j)}$ is the residual of the $j$th data constraint. The reweighting runs up to four iterations.

### D. Optional Step for Dense Flow Recovery

An independent flow computation system suffers the aperture problem in textureless regions, which limit its extension to dense motion estimation. One can address the problem by regularizing the system with smoothness constraints, as proposed in [9]. However, optimizing such a regularized objective

[6] More discussions on the behavior of $m$ are presented in our previous work [26].


Fig. 3. Middlebury benchmark sequences used for evaluation in this paper [1]. In all experiments, we use the gray-scale version.

functional is fundamentally different from optimizing an independent objective functional [27]. In this paper, we provide a more flexible alternative without altering the structure of our optimization process. We design an optional step, the employment of which is up to the application requirement. In particular, between $\boldsymbol{X}^{l,s}$ and $\boldsymbol{X}^{l,s+1}$, we insert an intermediate flow $\boldsymbol{X}^{l,s+\frac{1}{2}}$, computed by the following external diffusion step,

$$\boldsymbol{X}^{l,s+\frac{1}{2}} = \frac{\sum_{i\in \mathrm{N}} \omega_i \boldsymbol{X}_i^{l,s}}{\sum_{i\in \mathrm{N}} \omega_i} \tag{15}$$

where N is the set of the eight-points of the Compass Rose that the pixel is associated with; $\boldsymbol{X}_i^{l,s}$ is the flow vector obtained between (12) and (13) [or (14)] at the $i$th entry of N. We define

$$\omega_i = \begin{cases} \frac{1}{|E-E(i)|} & \text{if } E \neq E(i), \\ 1 & \text{otherwise,} \end{cases} \quad i \in \mathrm{N} \tag{16}$$

to steer the diffusion process toward pixels that have a consistent pattern. Equation (15) diffuses the flow to an aperture region from outside. If this step is selected, $\boldsymbol{X}^{l,s+\frac{1}{2}}$ is used as the current flow vector to warp the $N \times N$ surrounding patch centered at $(x + u^{l,s}, y + v^{l,s}, t + 1)$ in the second image. $A^{l,s+1}$ and $\boldsymbol{b}^{l,s+1}$ in (12) are then computed accordingly.

## IV. Experimental Results

In Section III-A, we have introduced an independent optical flow computation system, which consists purely of data constraints that are formed on the invariance of the proposed Compass Rose signature. This is our baseline algorithm. Our first experiment applies it to sparse feature tracking and compares its performance to two widely employed feature tracking systems. In Section III-C, we have suggested improving the robustness of our baseline system using adaptive solution. This improvement is tested and studied by the second experiment. Moreover, we compare this adaptive system with three state-of-the-art methods. Additionally, in Section III-D, we have designed an optional diffusion step: One can run it to overcome the aperture problem if necessary and skip it otherwise. Our third experiment activates this diffusion step on top of the second experiment and quantitatively evaluates the recovered dense flow field. Finally, we qualitatively analyze the application of Compass Rose to human motion estimation.

The first two experiments are conducted on the Middlebury training sequences (Fig. 3) [1]: *RubberWhale*, *Hydrangea*, *Grove2*, *Dimetrodon*, and *Venus*, which are in general well textured but contain challenging factors such as large displacement, substantial motion discontinuity, occlusion, disocclusion, nonrigid motion. Therefore, these sequences can provide abundant interesting points to test the feature tracking performance under a variety of difficult conditions. Five real scene MIT sequences (Fig. 9) [28]: *cameramotion*, *fish*, *hand*, *table*, and *toy*, with annotated ground truth optical flow,[7] are also used in the second experiment to evaluate the potential of Compass Rose for real applications. The third experiment is conducted on the UCL benchmark dataset created by Adoha *et al.* [29], which contain significant rotation and large textureless areas. Thus, they are suitable for testing the rotational robustness of Compass Rose, using the optional diffusion step to overcome the aperture problem. The forth experiment is conducted on a *HumanEva-II* benchmark human motion sequence, which contains fast articulated foot motion [30]. In all experiments (except one that runs the code of [31]), the test sequences are two-frame eight-bit gray scale.

Implementation parameters that have been described in Section III-B are set as the description and remain identical for all experiments. The flow computation accuracy is quantified by the average endpoint error (AEP) [1]

$$\frac{\sum_{i\in\mathcal{L}} \sqrt{(u_i^o - u_i)^2 + (v_i^o - v_i)^2}}{\#(\mathcal{L})} \tag{17}$$

and average angular error (AAE) [3]

$$\frac{\sum_{i\in\mathcal{L}} \arccos \frac{u_i^o u_i + v_i^o v_i + 1}{\sqrt{((u_i^o)^2+(v_i^o)^2+1)(u_i^2+v_i^2+1)}}}{\#(\mathcal{L})} \tag{18}$$

where $(u_i^o, v_i^o)$ and $(u_i, v_i)$ denote the computed flow and the ground truth flow of the $i$th feature in the tracking list $\mathcal{L}$, respectively. For dense flow computation, $\mathcal{L}$ is the whole image domain excluding pixels whose ground truth flow vectors are not available.

### A. Baseline Flow Computation Based on Compass Rose

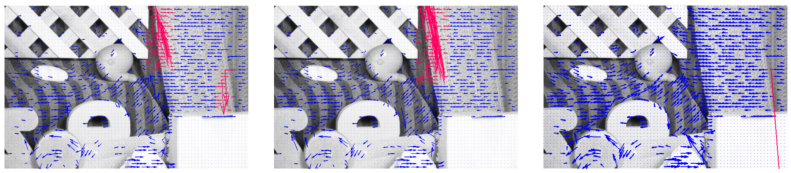
To evaluate the effectiveness of Compass Rose, we apply our baseline independent flow computation scheme (without adaptive solution or diffusion, Section III-A) to sparse feature tracking. Particularly, we compute the optical flow of the 25% of pixels that have the largest local variation, which is measured by the minor eigenvalue of the local structure tensor

$$\begin{bmatrix} \sum_G E_x^2 & \sum_G E_x E_y \\ \sum_G E_x E_y & \sum_G E_y^2 \end{bmatrix}$$

where $G$ is the local $5 \times 5$ window. Such a feature selection criterion is typical in practice (e.g., OpenCV [17]).

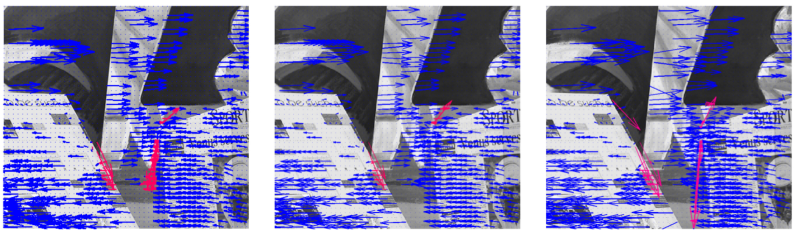
Under identical experimental settings (including using the same selected pixels, the same Gaussian pyramid constructed as in Section III-B, and solving all flow computation systems by their LSEs), we compare the effectiveness of the proposed Compass Rose with the two other signatures: intensity and gradient. This actually compares the proposed system with

[7]Sequences available at http://people.csail.mit.edu/celiu/motionAnnotation/



Lucas-Kanade (intensity) Uras et al. (gradient) proposed (Compass Rose)

Fig. 4. Sparse flow vectors recovered on *RubberWhale* by using Compass Rose (proposed), intensity (Lucas–Kanade), and gradient (Uras *et al.*) as the invariant signature to form data constraints. Red arrows indicate the flow vectors that are erroneously recovered. Compass Rose tracks the challenging feature points with satisfactory performance. Arrows are scaled uniformly for clear presentation.



Lucas-Kanade (intensity) Uras et al. (gradient) proposed (Compass Rose)

Fig. 5. Sparse flow vectors recovered on *Venus* by using Compass Rose (proposed), intensity (Lucas–Kanade), and gradient (Uras *et al.*) to form data constraints. Red arrows indicate the flow vectors that are erroneously recovered. All signatures suffer recovery errors around structural motion boundaries. Outliers cause more severe distortion to Compass Rose. Arrows are scaled uniformly for clear presentation.

Lucas–Kanade's and Uras–Girosi–Verri–Torre's computation. The first three lines of Table I compare the accuracy obtained by baseline Compass Rose (referred to as Baseline CR) to intensity and gradient signatures. Except on sequence *Venus*, Compass Rose yields higher accuracy than the other two signatures.

The most significant difference is observed on *RubberWhale*, as shown in Fig. 4. On this sequence, Lucas–Kanade and Uras–Girosi–Verri–Torre methods suffer large errors around the curtain boundary, which is a motion boundary surrounded by repetitive textures. In such an area, if the signature is not distinctive enough, i.e., if the spatial variation of the local image pattern is not well captured, then a small distortion in the temporal variation introduces large errors in the recovered flow. Therefore, the intensity and gradient signatures are not able to track these features accurately. In contrast, the Compass Rose signature takes the intrinsic variation of the repetitive pattern in 8 directions, and hence it captures the pattern more faithfully. Consequently, the distortion in the temporal variation is limited, and the flow vectors can still be consistently recovered.

If the motion boundary is structural rather than textured, all signatures suffer obvious errors around the boundary, but the problem may be more severe for Compass Rose. Fig. 5 demonstrates an example on *Venus*. This is because the Compass Rose has more entries, and hence one motion outlier gives rise to more distortion to our system. This problem can be addressed by the adaptive solution strategy, the evaluation of which is introduced in next subsection.

### B. Adaptive Solution

In Section III-C, we have suggested an adaptive solution approach to improve flow computation robustness at motion boundaries. Briefly, this approach estimates the presence of outliers by a measure $m \in [0, 1]$. If $m$ is larger than a predefined threshold $T$, the system is solved by the LSE solution, otherwise it is solved by the LAE solution. This is because the LAE solution is generally more robust to the LSE solution in the presence of outliers, but its computational cost is several times higher than the LSE. For example, our LAE is obtained by five iterations of LSE (one initial LSE and four reweighted LSE). This means that the computational load for an LAE solution is four times more than the computational cost for an LSE solution at a pixel.[8]

To find the most practical value of $T$, we investigate the flow computation performance of our system at various $T$ values on all test sequences. The computation accuracy on *RubberWhale* and *Venus* is observed to be sensitive to the change of $T$, whereas the computation accuracy on *Hydrangea*, *Grove2*, and *Dimetrodon* remains steady when $T$ changes. Fig. 6 plots the AAE results on *RubberWhale* and *Venus* at $T = 0, 0.1, \cdots, 1$. Each curve shows that, before convergence, the AAE always decreases when $T$ decreases. The observation on AEP performance is similar.

Fig. 7 marks where the angular error decreases most significantly (more than 1.5°) when $T$ decreases from 1 to 0.5. It can been seen that the most improved flow vectors are around the motion or intensity boundaries. This figure verifies that: 1) our inconsistency measure $m$ (Section III-C) well locates the presence of inconsistency, and 2) in regions with inconsistent motion or intensity patterns, the LAE solutions are on average more accurate than the LSE solutions.

Based on these observations, we suggest setting $T = 0.5$ in practice. The last line of Table I lists the numerical

[8]The precise CPU time depends on the size of the local region $N \times N$, the numerical scheme that solves the system and the CPU speed. By our Intel i7-3770 3.40 GHz processor, solving the LSE and LAE of a Compass Rose flow computation system constructed on a $7 \times 7$ patch takes averagely 0.024 ms and 0.086 ms, respectively.

TABLE I

AAE AND AEP PERFORMANCE OF INDEPENDENT FLOW COMPUTATION ON THE SELECTED 25% OF PIXELS BY: 1) LUCAS--KANADE SYSTEM [32], REFERRED TO AS INTENSITY; 2) URAS *et al.*'S SYSTEM [2], REFERRED TO AS GRADIENT; 3) PROPOSED BASELINE SYSTEM, REFERRED TO AS BASELINE CR; 4) PROPOSED SYSTEM WITH ADAPTIVE SOLUTION, REFERRED TO AS BASELINE CR + AS

| Method | Average | | Venus | | Grove2 | | RubberWhale | | Dimetrodon | | Hydrangea | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AAE | AEP | AAE | AEP | AAE | AEP | AAE | AEP | AAE | AEP | AAE | AEP |
| Intensity [32] | 5.82 | 0.34 | 7.39 | **0.41** | 4.08 | 0.28 | 8.52 | 0.37 | 2.33 | 0.13 | 6.78 | 0.52 |
| Gradient [2] | 5.78 | 0.34 | 8.01 | 0.43 | 4.00 | 0.28 | 8.76 | 0.69 | 1.89 | 0.10 | 6.24 | 0.48 |
| Baseline CR | 4.96 | 0.32 | 7.63 | 0.50 | 3.79 | 0.27 | 5.80 | 0.29 | 1.91 | 0.11 | 5.65 | 0.44 |
| Baseline CR+AS | **4.60** | **0.27** | **7.10** | 0.43 | **3.53** | **0.25** | **5.14** | **0.18** | **1.76** | **0.09** | **5.49** | **0.42** |

Bold numbers indicate the best performance.

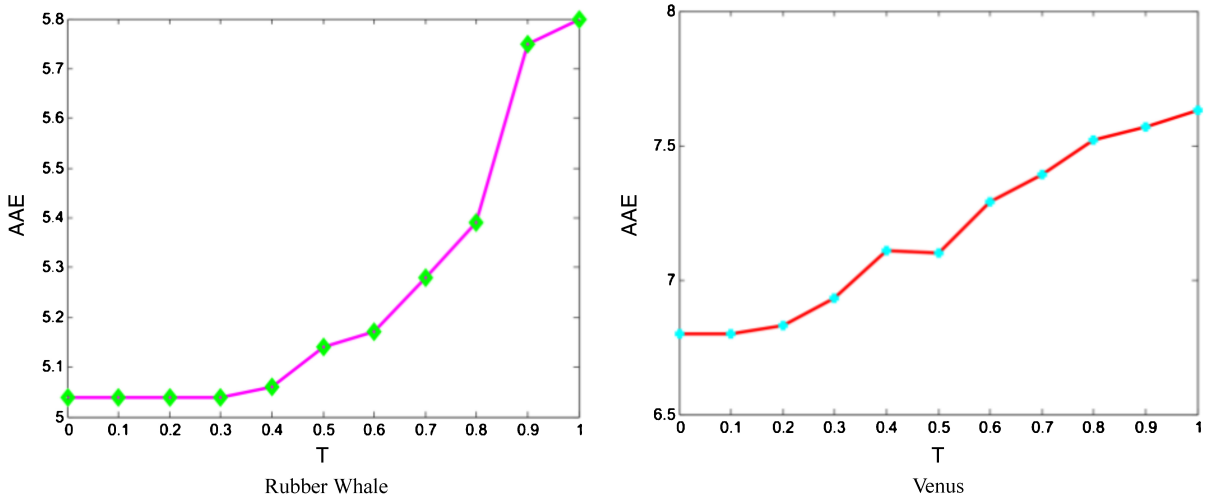



Fig. 6. AAE performance of our flow computation on *RubberWhale* and *Venus* under different values of $T$, the threshold for adaptive solution. On both sequences, the AAE is observed to decrease when $T$ decreases. However, as more flow vectors recovered by the LAE solution, the computation cost increases. In our implementation, an LAE solution incurs four times more computational cost than an LSE solution. Note that each curve depicts the change of AAE with $T$ on one image. Since there is no comparison between the two curves, their vertical axes have different scales and origins.

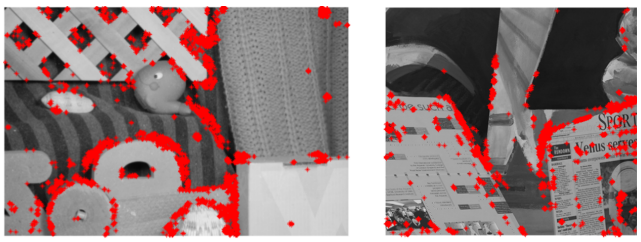



Fig. 7. Visualization of the benefits brought by using adaptive solution to *Venus* and *RubberWhale*. Red color indicates where the flow angular error decreases most significantly ($> 1.5^\circ$) when $T$, the adaptive solution threshold, decreases from 1.0 to 0.5. This shows that the adaptive solution approach benefits flow computation around motion and intensity boundaries.

performance of our system with $T = 0.5$ (Baseline CR+AS). It achieves the highest accuracy on all test sequences. Fig. 8 visualizes the ground truth flow vectors $(u^o, v^o)$, the recovered flow vectors $(u, v)$, and the difference vectors $(u - u^o, v - v^o)$ at samples of the selected 25% of pixels.

As sparse feature tracking is favored by real-time applications (e.g., human–machine interaction [12], structure from motion for image-based rendering [13]), we evaluate the runtime of the Baseline CR+AS system on five MIT sequences (Fig. 9). On the central frame of each sequence, we select more than 2000 feature points by the strategy suggested in OpenCV [17].[9] The speed of Baseline CR+AS tracking of each feature varies with the values of $L$, $S$, $N$, the number of IRLS iterations, and the numerical method that solves (12). With the parameters set as in Section III-B, our tracking is implemented in MATLAB by a mex function, which calls a linear algebra package (LAPACK) routine dgesv to solve $(A^{l,s})^T A^{l,s} \delta \boldsymbol{X}^{l,s} = (A^{l,s})^T \boldsymbol{b}^{l,s}$ for $\delta \boldsymbol{X}^{l,s}$. Table II displays the tracking time and the associated accuracy of the Baseline CR+AS system on an Intel i7-3770 3.40 GHz processor with 8 GB RAM.

For reference, we compute the flow of the selected features by three state-of-the-art methods: large displacement optical flow (LDOF) [31], Classic+NL-fast [23], and SIFT-Flow [5], by running their publicly available MATLAB code on the same machine mentioned above.[10] Table II presents their

[9]In terms of [17], we retain the pixels that have a $\lambda_m$ larger than $k\lambda_{\max}$. After nonmaximum suppression, we keep the subset of those pixels so that the minimum distance between any pair of pixels is larger than 3 pixels. Here, $k$ varies with the sequence nature, so as to select more than 2000 feature points.

[10]The code we use for LDOF, Classic+NL-fast and SIFT-Flow is downloaded at http://lmb.informatik.uni-freiburg.de/resources/binaries/pami2010Matlab.zip, http://cs.brown.edu/~dqsun/code/cvpr10_flow_code.zip, and http://people.csail.mit.edu/celiu/ECCV2008/, respectively. All parameters are set by default.

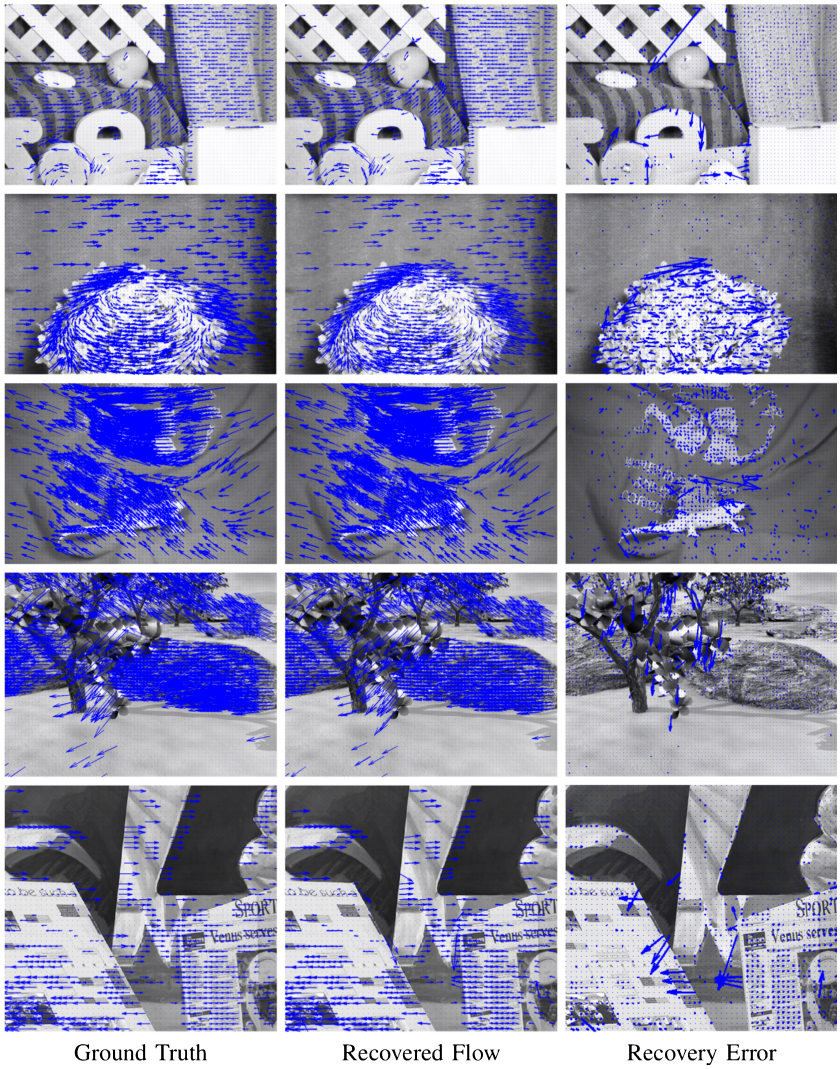



Fig. 8. Results by the proposed independent flow computation system on Middlebury training sequences. Left and center columns: the ground truth flow $(u^o, v^o)$ and computed flow $(u, v)$ at the selected 25% of pixels, respectively, scaled uniformly and displayed at an interval of 8 pixels. Right column: the difference vectors between the computed flow vectors and the ground truth, i.e., $(u - u^o, v - v^o)$; arrows in each image are enlarged uniformly. See Table I for quantitative evaluation.


Fig. 9. Test frames of the MIT sequences. Testing of our method and Classic+NL-fast is on the gray-scale version. Testing of LDOF is on the RGB version.

runtime and the average accuracy of the flow estimated at the selected points.[11] On average, the endpoint error of our method is 0.13 pixels higher than LDOF and 0.2 pixels higher than Classic+NL-fast at each feature point, but due to the regularization used by LDOF and Classic+NL-fast, it is tens of seconds faster. This advantage becomes more obvious on higher resolution images. Similar to our method, SIFT-Flow uses the invariance of a local descriptor to form the data constraint for flow estimation. As it is proposed for scene matching, SIFT-Flow has higher requirement on the speed than the accuracy. It runs several seconds faster than our method on *camera*, *fish*, and *table*; but on images with higher resolution such as *hand* and *toy*, the two methods have similar computation time. On average, the endpoint error of SIFT-Flow at each feature point is 0.2 pixels higher than the Baseline CR+AS. Thus, our method is a favorable compromise between speed of SIFT-Flow and the accuracy of classic optical flow methods.

It should be noted that the speed comparison is not precise. For any of the four methods, the runtime depends on the coding, which can be further optimized. For example, the

[11] On each sequence, for each method, the average error of the flow estimated at the selected features is lower than the average error of the whole flow field.

TABLE II

AVERAGE ACCURACY (MEASURED BY AEP) AND TOTAL RUNTIME OF THE BASELINE CR+AS SYSTEM, LDOF, CLASSIC-NL-FAST, AND SIFT-FLOW METHODS ON THE CENTRAL TWO FRAMES OF EACH MIT SEQUENCE

| | | LDOF | | Classic+NL-fast | | SIFT-Flow | | Baseline CR+AS | |
|---|---|---|---|---|---|---|---|---|---|
| sequence | Feature No. | AEP | Time (sec) | AEP | Time (sec) | AEP | Time (Sec) | AEP | Time (Sec) |
| camera | 2087 | 0.11 | 17.53 | 0.17 | 65.04 | 0.37 | 9.11 | 0.21 | 17.47 |
| fish | 2412 | 0.28 | 65.19 | 0.42 | 65.97 | 0.63 | 9.22 | 0.63 | 16.71 |
| hand | 2294 | 1.31 | 54.81 | 0.79 | 92.49 | 1.59 | 15.17 | 1.22 | 14.67 |
| table | 2243 | 0.50 | 28.82 | 0.48 | 61.18 | 0.86 | 10.35 | 0.64 | 13.91 |
| toy | 2917 | 0.25 | 44.32 | 0.22 | 151.83 | 0.60 | 21.43 | 0.36 | 17.94 |

TABLE III

AAE AND AEP PERFORMANCE OF JOINT FLOW COMPUTATION WITH THE PROPOSED SIGNATURE (ABBREVIATED AS CR) AND THE GRADIENT VECTOR AS THE INVARIANT SIGNATURE TO FORM DATA CONSTRAINTS. BOLD LETTERS INDICATE THE NAMES OF SEQUENCES ON WHICH COMPASS ROSE PERFORMS BETTER. PROPOSED SIGNATURE SIGNIFICANTLY OUTPERFORMS THE GRADIENT VECTOR

| Method | AAE | AEP | AAE | AEP | AAE | AEP | AAE | AEP | AAE | AEP | AAE | AEP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Average** | | **blow1Txtr1** | | **blow19Txtr2** | | **Brickbox1t1** | | Brickbox2t2 | | **Crates1** | |
| Gradient | 11.47 | 4.20 | 5.62 | 0.22 | 9.31 | 1.13 | 14.93 | 4.44 | 21.98 | 13.30 | 15.57 | 5.46 |
| CR | 8.91 | 3.11 | 3.29 | 0.14 | 6.46 | 0.60 | 3.56 | 1.14 | 30.98 | 13.99 | 10.00 | 4.11 |
| | **Crates1Htxtr2** | | **Crates2** | | **Crates2Htxtr1** | | **drop1Txtr1** | | **drop9Txtr2** | | GrassSky0 | |
| Gradient | 9.80 | 2.56 | 31.00 | 19.98 | 18.88 | 14.29 | 7.93 | 0.66 | 20.71 | 7.87 | 10.24 | 3.78 |
| CR | 5.31 | 1.85 | 17.99 | 11.31 | 10.76 | 10.03 | 4.77 | 0.16 | 17.01 | 3.99 | 10.42 | 4.78 |
| | **GrassSky9** | | **GroveSun** | | Mayan1 | | **Mayan2** | | **Robot** | | **roll1Txtr1** | |
| Gradient | 4.42 | 1.27 | 9.16 | 0.59 | 6.48 | 1.37 | 5.50 | 0.79 | 13.37 | 4.73 | 0.37 | 0.01 |
| CR | 2.33 | 0.85 | 8.77 | 0.48 | 10.90 | 2.93 | 3.46 | 0.72 | 12.78 | 3.36 | 0.28 | 0.00 |
| | **roll9Txtr2** | | **Sponza1** | | **Sponza2** | | **street1Txtr1** | | **TxtLMovement** | | **TxtRMovement** | |
| Gradient | 3.39 | 0.16 | 17.25 | 2.31 | 14.63 | 1.95 | 19.41 | 7.72 | 2.67 | 1.18 | 1.25 | 1.05 |
| CR | 1.91 | 0.05 | 14.26 | 1.97 | 10.38 | 1.47 | 17.58 | 6.56 | 0.87 | 0.57 | 0.96 | 0.62 |

Baseline CR+AS system can solve (12) by a faster numerical scheme such as Elmroth–Gustavson's algorithm, which solves a linear system up to four times faster than the LAPACK routine [33]. Also, subsequent work on LDOF has shown that it runs significantly faster using a GPU implementation [34]. As the Compass Rose tracking of each feature is independent, the tracking of all features can also be parallelized, and substantially accelerated using a multicore CPU, GPU, or integrated circuits such as field programmable gate array.

### C. Optional Diffusion Step for Joint Flow Computation

On top of the independent Compass Rose flow computation tested in previous section, we activate the optional diffusion step designed in Section III-D, with the threshold $T$ for adaptive solution fixed to 0.5. We use the UCL benchmark sequences to test the performance of Compass Rose flow computation under fast rotation, with the aperture problem addressed by the optional diffusion step. Fig. 10 illustrates five example sequences and the corresponding recovery results by our method. With identical experimental settings, we replace the data system in our joint flow computation with Uras *et al.*'s system. This compares the effectiveness of using Compass Rose and the gradient vector as the invariant signature to recover fast rotation. The recovered flow fields on the five example sequences are visualized by the third row of Fig. 10. The AAE and AEP results obtained by both systems on all 23 experimental sequences are listed in Table III. The Compass Rose signature outperforms the gradient vector on 20 out of the 23 sequences. On *Brickbox2t2*, *GrassSky0*, *Mayan1* (note that *GrassSky0* and *Mayan1* depict the same motion), Compass Rose has inferior performance in areas that contain large disocclusion, which disturbs the estimation of motion boundary direction and the directional derivative entries of the signature. However, as this situation is relatively rare, Compass Rose yields notably higher accuracy on average. Especially on sequences that contain fast rotation, such as *Brickbox1t1*, *Crates2*, *drop9Txtr2*, the improvement is significant (see Fig. 10 for visual comparison).

### D. Human Motion Estimation

Finally, we apply our joint optical flow computation to human motion estimation, which generally involves articulated fast rotation. Fig. 11 illustrates an experiment conducted on test sequence *Running* from the benchmark human motion dataset *HumanEva-II* [30]. Same as previous works, our experiment only uses frame 546 and frame 550. Many state-of-the-art works use this pair of frames to evaluate how well an optical flow computation algorithm can handle large displacement, as the right leg and foot of the running person undergo challenging fast motion.

Our flow recovery result demonstrates that the diffusion step overcomes the aperture problem (Fig. 11). Moreover, different moving parts are well recovered with consistent motion. One can clearly distinguish the motion of each body segment. Compared to the results reported on the same test frames [35] (and the results reported in [36]), the presented method recovers motion more accurately in the leg and

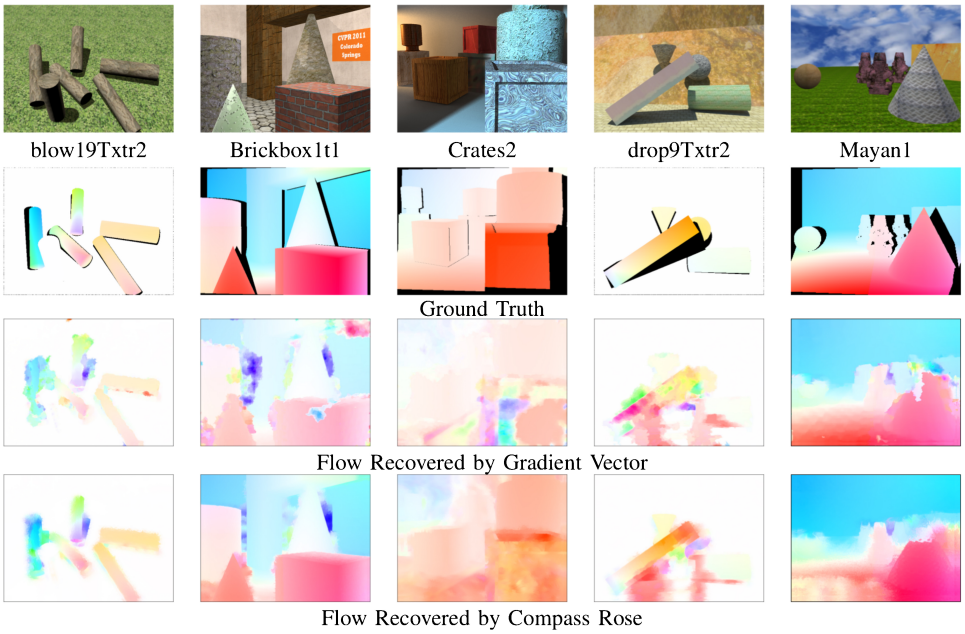



Fig. 10. Comparison between using the Compass Rose signature and the gradient vector signature in the framework of joint flow computation on UCL benchmark sequences [29]. Flow recovery is visualized by the Middlebury colorcoding scheme. Implementation is conducted on the gray-scale version of the test images. Refer to Table III for quantitative evaluation.

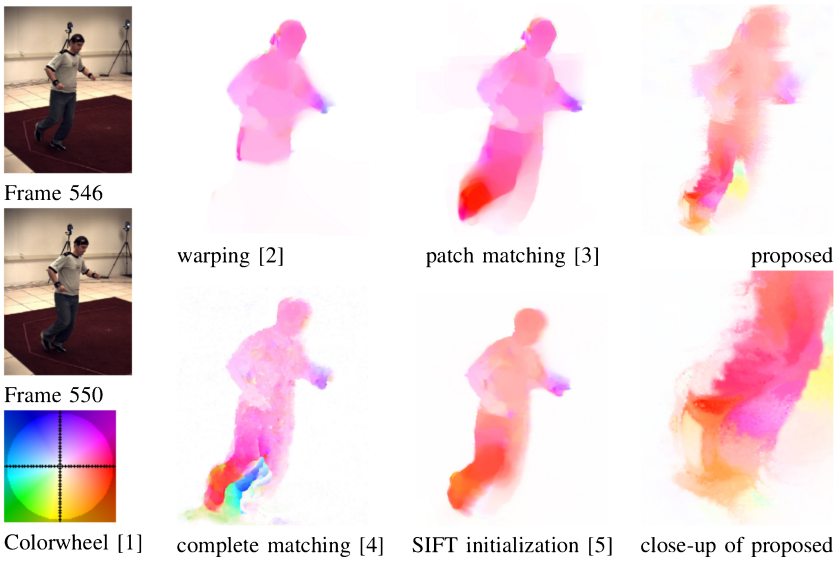



Fig. 11. Test results on the frame 546 and 550 from a benchmark *HumanEva-II* sequence by previous methods and our algorithm. Results by previous methods are taken from the corresponding publications. As different work uses a different colorwheel, we adjust the pictures by unifying the colorwheel for a clear comparison.

foot area, whereas other works may perform better in the head and left arm area. Due to the lack of quantitative ground truth, the performance difference cannot be measured numerically.

As Compass Rose is designed to recover fast rotation, we focus the comparison on the right leg and foot. Their displacement is too large to be captured by [35] and [36], which use gradient constancy as the data constraints. Presumably, this is because the objective functionals of [35] and [36] emphasize the smoothness constraints, which benefit consistency but smooths out fast motion. To address this problem, [37] and [38] employ data matching to enhance the data constraints. More specifically, [37] uses exhaustive searching and matching in a restricted area; and [38] initializes the flow filed in each refinement level by SIFT matching.

We observe that under some types of fast rotation such as the right foot's motion depicted in this scene, although the resultant displacement is large, the neighboring flow vectors are well connected by an affine motion model, rather than the smoothness constraints. With an appropriate rotational robust signature to form data constraints, the motion pattern can be recovered even without matching. This observation is supported by our experimental result, which is comparable to [37], in the sense that [37] recovers the shape of the right foot more faithfully, whereas our system keeps the boundaries of the right leg more sharply.

## V. Conclusion

In this paper, we have introduced the Compass Rose signature for optical flow computation. The signature was motivated by the need for an efficient, differentiable signature that is robust to high amounts of local rotation and translation. It has been shown that it at least equals image signatures traditionally used for flow computation, and exceeds them in the challenging conditions for which it was designed. We have demonstrated its use in both solving for a sparse independent set of feature correspondences, and for dense flow field calculation, using image sequences that include sharp motion boundaries, featureless regions, and fast motion.

## References


[1] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *Int. J. Comput. Vision*, vol. 92, no. 1, pp. 1–31, 2011.

[2] S. Uras, F. Girosi, A. Verri, and V. Torre, "A computational approach to motion perception," *Biol. Cybern*, vol. 60, no. 2, pp. 79–87, 1988.

[3] J. Barron, D. J. Fleet, and S. Beauchemin, "Performance of optical flow techniques," *Int. J. Comput. Vision*, vol. 12, no. 1, pp. 43–77, 1994.

[4] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[5] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. Freeman, "SIFT flow: Dense correspondence across different scenes," in *Proc. Eur. Conf. Comput. Vision*, 2008, pp. III: 28–42.

[6] E. Tola, V. Lepetit, and P. Fua, "DAISY: An efficient dense descriptor applied to wide-baseline stereo," *IEEE Trans. PAMI*, vol. 32, no. 5, pp. 815–830, May 2010.

[7] M. Spivak, *A Comprehensive Introduction to Differential Geometry*. Berkeley, CA, USA: Publish or Perish, Inc., 1979.

[8] Y. Niu, A. Dick, and M. Brooks, "Locally oriented optical flow computation," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1573–1586, Apr. 2012.

[9] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods," *Int. J. Comput. Vision*, vol. 61, no. 3, pp. 211–231, 2005.

[10] Y. B. Sirotin and A. Das, "Anticipatory haemodynamic signals in sensory cortex not predicted by local neuronal activity," *Nature*, vol. 457, pp. 475–479, Jan. 2009.

[11] R. Messing, C. Pal, and H. Kautz, "Activity recognition using the velocity histories of tracked keypoints," in *Proc. IEEE Int. Conf. Comput. Vision*, Jul. 2009, pp. 104–111.

[12] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Proc. Eur. Conf. Comput. Vision*, 2006, pp. II: 428–441.

[13] v. d. H. Anton, A. Dick, T. Thorsten, B. Ward, and P. H. S. Torr, "Videotrace: Rapid interactive scene modelling from video," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 86-1–86-6, 2007.

[14] J. Shi and C. Tomasi, "Good features to track," in *Proc. Comput. Vision Pattern Recognit.*, 1994, pp. 593–600.

[15] R. Szeliski, *Computer Vision: Algorithms and Applications*. Berlin, Germany: Springer, 2011.

[16] R. Laganière, *OpenCV 2 Computer Vision Application Programming Cookbook*. Birmingham, U.K.: Packet Publishing, 2011.

[17] J.-Y. Bouguet, *Pyramidal Implementation of the Lucas–Kanade Feature Tracker*, OpenCV documentation, Intel Corporation, 2000.

[18] C. E. Shannon, "Communication in the presence of noise," *Proc. IEEE*, vol. 86, no. 2, pp. 10–21, Nov. 2012 (reprint as classic paper).

[19] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. Englewood Cliffs, NJ, USA: Prentice Hall, 2006.

[20] R. Cipolla and A. Blake, "Image divergence and deformation from closed curves," *Int. J. Robot. Res.*, vol. 16, no. 1, pp. 77–96, Feb. 1997.

[21] M. Campani and A. Verri, "Computing optical flow from an overconstrained system of linear algebraic equations," in *Proc. Int. Conf. Comput. Vision*, 1990, pp. 22–26.

[22] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.

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

[24] H. Anton and C. Rorres, *Elementary Linear Algebra: Applications Version*, 7th ed. New York, NY, USA: Wiley, 1994.

[25] G. Strang, *Linear Algebra and Its Applications*, 1st ed. San Francisco, CA, USA: Academic Press, 1976.

[26] Y. Niu, A. Dick, and M. Brooks, "A new inconsistency measure for linear systems and two applications in motion analysis," in *Proc. Image Vision Comput.*, 2009, pp. 12–17.

[27] S. Birchfield and S. Pundlik, "Joint tracking of features and edges," in *Proc. Comput. Vision Pattern Recognit.*, 2008, pp. 1–6.

[28] C. Liu, W. Freeman, E. Adelson, and Y. Weiss, "Human-assisted motion annotation," in *Proc. Comput. Vision Pattern Recognit.*, 2008, pp. 1–8.

[29] O. M. Adoha, A. Humayun, M. Pollefeys, and J. Brostow, "Learning a confidence measure for optical flow," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 5, pp. 1107–1120, Aug. 2012.

[30] M. J. Black *et al.*, (2007). *HumanEva* [Online]. Available: http://vision.cs.brown.edu/humaneva/.

[31] T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," *IEEE. Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 3, pp. 500–513, Mar. 2011.

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

[33] E. Elmroth and F. Gastavson, "A faster and simpler recursive algorithm for the lapack routine dgels," *BIT Numerical Math.*, vol. 41, no. 5, pp. 936–949, 2001.

[34] N. Sundaram, T. Brox, and K. Keutzer, "Dense point trajectories by gpu-accelerated large displacement optical flow," in *Proc. Eur. Conf. Comput. Vision*, 2010, pp. 438–451.

[35] T. Brox, C. Bregler, and J. Malik, "Large displacement optical flow," in *Proc. Comput. Vision Pattern Recognit.*, 2009, pp. 41–48.

[36] 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. Vision*, 2004, pp. 25–36.

[37] F. Steinbruecker, T. Pock, and D. Cremers, "Large displacement optical flow computation without warping," in *Proc. Int. Conf. Comput. Vision*, 2009, pp. 1609–1614.

[38] L. Xu, J. Jia, and Y. Matsushita, "Motion detail preserving optical flow estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1744–1757, Sep. 2012.


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

She is currently a Lecturer at the College of Computer Science and Technology, Jilin University, China. Her current 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,

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

**Michael Brooks** received the Ph.D. degree in computer vision from the University of Essex, Colchester, U.K., in 1983.

He is currently a Professor of computer science at the University of Adelaide, Adelaide, Australia. Since 1983, he has focused on a range of problems including shape from shading, stereo, structure from motion, and visual surveillance.

Prof. Brooks is a fellow of the Australian Academy of Technological Sciences and Engineering, Melbourne, Australia. and the Australian Computer Society, Sydney, Australia, and serves on the Board of National ICT Australia, Sydney, Australia. He currently holds the position of Deputy Vice Chancellor (Research) at the University of Adelaide.