Int J Comput Vis (2011) 92: 53–70
DOI 10.1007/s11263-010-0402-2

A Local Algorithm for the Computation of Image Velocity via Constructive Interference of Global Fourier Components

Babette Dellen · Florentin Wörgötter

Received: 30 January 2009 / Accepted: 12 October 2010 / Published online: 3 November 2010

Abstract A novel Fourier-based technique for local motion detection from image sequences is proposed. In this method, the instantaneous velocities of local image points are inferred directly from the global 3D Fourier components of the image sequence. This is done by selecting those velocities for which the superposition of the corresponding Fourier gratings leads to constructive interference at the image point. Hence, image velocities can be assigned locally even though position is computed from the phases and amplitudes of global Fourier components (spanning the whole image sequence) that have been filtered based on the motion-constraint equation, reducing certain aperture effects typically arising from windowing in other methods. Regularization is introduced for sequences having smooth flow fields. Aperture effects and their effect on optic-flow regularization are investigated in this context. The algorithm is tested on both synthetic and real image sequences and the results are compared to those of other local methods. Finally, we show that other motion features, i.e. motion direction, can be computed using the same algorithmic framework without requiring an intermediate representation of local velocity, which is an important characteristic of the proposed method.

Keywords Image velocity · Local methods · Fourier transformation · Constructive interference

B. Dellen (✉)
Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Llorens i Artigas 4-6, 08028 Barcelona, Spain
e-mail: bdellen@iri.upc.edu

F. Wörgötter
Bernstein Center for Computational Neuroscience Göttingen, III. Physikalisches Institut-Biophysik, Georg-August-University Göttingen, Friedrich-Hund Platz 1, 37077 Göttingen, Germany

1 Introduction

The computation of optic flow is a prerequisite for solving many important tasks encountered in the fields of computer vision and robotics, e.g., navigation, surveillance, and tracking. As a consequence, many methods for optic flow have been proposed in the past (for reviews see Barron et al. 1994; Galvin et al. 1998; Baker et al. 2007). We distinguish roughly between local and global methods. Local methods for optic-flow computation perform a local analysis of the time-dependent pixel intensities inside a small measurement window and are as such concerned with the basic principles of local velocity measurement. Common approaches are differential (Lucas and Kanade 1981; Nagel 1987), phase-based (Fleet and Jepson 1990; Gautama and Van Hulle 2003), energy-based, (Heeger 1987), and region-based-matching techniques (Anandan 1987; Singh 1990). Usually, local algorithms allow the computation of a confidence function, leading ultimately to non-dense optic-flow fields.

Global methods perform a global optimization of image velocities, usually through the formulation of a global cost function, which is then minimized, e.g., by using methods such as graph cuts or belief propagation (Boykov et al. 2001; Weiss and Freeman 2001). Global optimization has also been successfully implemented by using iterative solutions of a set of partial differential equations (Horn and Schunck 1981; Weickert and Schnörr 2001; Bruhn et al. 2005; Papenberg et al. 2006). Recently, there has been much progress along this strand of optic-flow research and extensive comparisons of algorithms have been provided for the Middlebury benchmark data set (Baker et al. 2007) (see also http://vision.middlebury.edu/flow/). However, basic methods of velocity measurements, i.e. local algorithms, are also important here, since global algorithms always contain a

local measurement as a core ingredient, on top of which global optimization is applied. However, while in many algorithms local and global computations can be separated (Cooke 2008; Felzenszwalb and Huttenlocher 2006), this is not always the case (Horn and Schunck 1981; Weickert and Schnörr 2001; Bruhn et al. 2005; Papenberg et al. 2006).

Estimation of motion has been explained in the past by formulating the problem in the Fourier domain (Adelson and Bergen 1985; Simoncelli 1993). Here, the motion of a translating 2D intensity image corresponds to a characteristic plane in the Fourier domain, defined through the motion-constraint equation, which relates the image velocity with the spatial and temporal frequencies of the Fourier space. The image velocity determines the orientation of the plane and can be derived from the nonzero Fourier components. The computations of local methods are restricted to a small spatiotemporal region of the image sequence (Barron et al. 1994; Galvin et al. 1998), i.e., in the Fourier space formulation, a *local* power spectrum is obtained by restricting the Fourier transformation to the size of the measurement window, e.g., energy-based methods using Gabor filters (Gaussian windows) (Adelson and Bergen 1985; Heeger 1987).[1] If this region is sufficiently small, it will, in the ideal case, contain only the object of interest, which will then have a unique velocity. There is a catch though: Since the measurement of frequencies is based on spatiotemporal sampling, the velocity estimates become less accurate with decreasing window size. This phenomenon is known as the uncertainty principle in signal processing (Wilson and Granlund 1984). As a consequence, local methods (Barron et al. 1994; Galvin et al. 1998) can only compute accurate velocity estimates at points which have sufficient image structure, such as edges. Global methods in some sense attempt to solve this problem by adjusting the window sizes locally via an iterative optimization process.

Previously, several methods have tried to facilitate motion processing by representing the image sequence in the *global* Fourier domain. Vernon developed a Fourier-based method for segmenting images which are composed of an occluding foreground and occluded background for situations in which the object velocity is constant and normal to the principal ray of the image sensor (Vernon 1998). Dellen et al. (2007) demonstrated that objects which are moving in opposing directions are separable in Fourier space. Global Fourier components have also been used to reduce motion estimation errors in a hybrid approach (Briassouli and Narendra 2006).

[1]These methods can be shown to be closely related and, under certain conditions, to be identical to differential methods (Lucas and Kanade 1981; Simoncelli 1993; Barron et al. 1994), matching techniques (Singh 1990; Anandan 1987), and phase-based methods (Fleet and Jepson 1990; Gautama and Van Hulle 2003).

In this paper, a novel approach for local velocity estimation based on processing in a global Fourier space is proposed. Global Fourier components that fulfill the motion-constraint equation for a certain test velocity are selected. The amplitudes and phases of the selected global Fourier components allow us to reconstruct spatiotemporal position from the signal, and the reconstruction can then be compared directly with the original image, constituting the essential idea of the proposed algorithm. Similar to region-based matching (Singh 1990; Anandan 1987), where the image of an adjacent frame is translated and the pixel intensities are compared with the original image, we compare our reconstructed image with the original one, which allows, at the read-out stage, applying similar strategies as region-based matching. However, applying spatiotemporal frequency filters in Fourier space instead of shifting pixels in real space has the following advantages: (i) Sub-pixel velocities are automatically defined in the framework. (ii) Aperture effects can be reduced because the whole sequence is considered when the motion-constraint equation is applied. (iii) As a consequence, multiple (transparent) motions can be treated in the framework. (iv) The motion-constraint equation can be relaxed without increasing the complexity of the algorithm. (v) The framework can be easily applied to the computation of other motion features by formulating appropriate frequency filters, e.g., for motion direction.

Following this introduction, in Sect. 2, we formulate the algorithm and demonstrate theoretically how local image velocity can be derived from global Fourier components using the principle of constructive interference. The inclusion of a regularization operation (without optimization) further allows the computation of dense smooth optic-flow fields and quantitative comparison with other local methods which intrinsically perform regularization through the application of a measurement window. In Sect. 3, we investigate the core parameters of the algorithm. The algorithm is then applied to several real image sequences and evaluated qualitatively and quantitatively. We demonstrate on the example of an artificial and a real sequence that the measurement-window-induced aperture problem is reduced in our method compared to others. In this context, we shortly discuss some fundamental effects of optic-flow regularization, i.e. smoothing and global optimization, which can be applied on top of the algorithm. We also analyze the behavior of the algorithm with respect to image properties such as pattern velocity, noise, and motion jitter. In this context, our method is compared with region-based matching. Finally, we show how the framework can be used to compute motion direction. In Sect. 4, the results are discussed and directions for future research are given.

Part of the results shown in this paper have been previously published in conference proceedings (Dellen and

Wörgötter 2008). Following parts have been added. We describe the algorithm in greater detail and theoretically investigate its response to local motion. We further included an algorithmic extension that allows transparent motion to be detected. A sensitivity analysis with respect to the core parameters of the algorithm has been included. Furthermore, to allow better comparison with other local methods, we introduced regularization delivering dense smoothed optic-flow fields, and compare the method in detail with region-based matching and also with other local methods. Quantitative results for some standard motion sequences are provided. We also show that the framework can be used to infer other local motion features, e.g. motion direction, directly from the image sequence, without requiring an intermediate representation of local velocity.

2 Algorithm

2.1 Basic Algorithmic Framework

The motion of a translating one-dimensional intensity image corresponds to a characteristic line pattern in the Fourier domain (Adelson and Bergen 1985; Simoncelli 1993), where the spatial frequency k and the temporal frequency k_t are related via the motion-constraint equation $kv = k_t$, where v is the velocity of the pattern. Similarly, a translating 2D intensity image represents a plane in Fourier space, which is defined by the respective motion-constraint equation $k_x v_x + k_y v_y = k_t$. This property can be used to measure optical flow by searching for the plane that best fits the power spectrum of the spatiotemporal signal. However, spatiotemporal position is also encoded in the amplitudes and phases of the Fourier components, which can be used to reconstruct spatiotemporal position from the signal. Hence, we can first select all Fourier components that fulfill the motion-constraint equation (for a given test velocity) and then reconstruct the image in real space. At a given local point, the Fourier components interfere with each other constructively only if the test velocity is close to the true local image velocity, and, as a consequence, more of the original image intensity can be recovered, which can be quantified.

On the basis of this idea, we formulate an algorithm for local-motion detection from global Fourier components as follows. Let the visual scene be represented by a three-dimensional discrete function of intensity values $I(\mathbf{x}, t)$, where $\mathbf{x} = (x, y)$ defines the spatial dimensions and t the temporal dimension. We further assume that the image sequence has a mean intensity of zero. Using Fourier decomposition, the image sequence can be described as a superposition of translating gratings, such that

$$I(\mathbf{x}, t) = \iiint_{-\infty}^{+\infty} A(\mathbf{k}, k_t)\cos(\mathbf{k}\mathbf{x} - k_t t) + B(\mathbf{k}, k_t)\sin(\mathbf{k}\mathbf{x} - k_t t)d\mathbf{k}dk_t, \quad (1)$$

where $\mathbf{k} = (k_x, k_y)$ is a wave vector with spatial frequencies k_x and k_y, and k_t is a temporal frequency. The amplitudes $A(\mathbf{k}, k_t)$ and $B(\mathbf{k}, k_t)$ depend on the spatial and temporal frequencies of the gratings and represent the image sequence in 3D Fourier space. Each grating moves with a velocity $\mathbf{v}_\perp = k_t\mathbf{k}/k^2$, where $k = |\mathbf{k}|$ is the absolute spatial frequency of the grating. The combined movements of the gratings contributing to a particular point $(\mathbf{x}, t)$ of the image sequence determine the local velocity $\mathbf{V}(\mathbf{x}, t)$ at this point. Importantly, for non-transparent conditions, only intensities belonging to a single object are represented at a local point $(\mathbf{x}, t)$. The contribution of a grating to the velocity of a point $(\mathbf{x}, t)$ of the image sequence can be quantified by assigning a weight to every grating. For example, a grating which has a negative amplitude at a point of positive intensity is contributing destructively, while a grating of positive amplitude contributes constructively. Note that the intensity $I(\mathbf{x}, t)$ has a mean intensity of zero and thus can also take negative values. Thus, weights depend not only on the amplitude of the grating at this point, but also on the intensity of the point itself, such that

$$w(\mathbf{x}, t, \mathbf{k}, k_t) = [A(\mathbf{k}, k_t)\cos(\mathbf{k}\mathbf{x} - k_t t) + B(\mathbf{k}, k_t)\sin(\mathbf{k}\mathbf{x} - k_t t)] \times \operatorname{sign}[I(\mathbf{x}, t)], \quad (2)$$

where $\operatorname{sign}[\pm a] = \pm 1$ is the sign function. Hence, for each point $(\mathbf{x}, t)$ of the image sequence, we obtain a set of weighted velocities $\{\mathbf{v}_\perp(\mathbf{k}, k_t); w(\mathbf{x}, t, \mathbf{k}, k_t)\}$. The set of weighted velocities can be used to estimate the local velocity $\mathbf{V}(\mathbf{x}, t)$ of the image point since all gratings belonging to an object moving with a velocity $\mathbf{v}$ fulfill the motion-constraint equation $k_t = \mathbf{v}\mathbf{k} = \mathbf{v}_\perp(\mathbf{k}, k_t)\mathbf{k}$ (Adelson and Bergen 1985).

We estimate the *local velocity* by employing the following voting scheme: Each component velocity votes with its assigned weight for all local velocities which lie along the corresponding constraint line, yielding a map of votes

$$m(\mathbf{U}, \mathbf{x}, t) = \iiint_{-\infty}^{+\infty} w(\mathbf{x}, t, \mathbf{k}, k_t) \times \exp[-(k_t - \mathbf{U}\mathbf{k})^2/(\xi|\mathbf{k}|)^2]d\mathbf{k}dk_t, \quad (3)$$

where the parameter ξ is the width of the Gaussian. The Gaussian function, which reaches its maximum at $k_t = \mathbf{U}\mathbf{k}$, implements the motion-constraint equation. If ξ is large, gratings with a combination of frequencies in the neighborhood of the selected velocity also contribute to the voting. If ξ is small, almost only those gratings contribute to the voting which fulfill the motion-constraint equation exactly and

the Gaussian converges to a delta function. A large ξ can be advantageous if the motion of the image point is accelerated and thus smeared in Fourier space. It should be noted however that accelerated motion is a violation of the motion constraint equation. If accelerated motion is of interest, an extended motion-constraint equation with additional higher order terms would be more accurate and make the estimation of acceleration feasible at the same time.

The map of votes peaks at $\mathbf{U} = \mathbf{V}(\mathbf{x}, t)$ if the velocity signal is sufficiently strong, where $\mathbf{V}(\mathbf{x}, t)$ is the local velocity of the point. Thus, we compute the velocity estimate $\mathbf{V_e}(\mathbf{x}, t)$ by finding the maximum of the map of votes and taking its argument

$$\mathbf{V_e}(\mathbf{x}, t) = \arg\{\max[m(\mathbf{U})]\}. \tag{4}$$

A schematic of the algorithm is presented in Fig. 1A–B.

2.2

We shortly illustrate that the algorithm indeed computes local velocity on the example of a moving dot. Let $I(\mathbf{x}, t) = \delta(\mathbf{x} - \mathbf{v}t)$, then, applying the 3D Fourier transform yields

$$F(\mathbf{k}, k_t) = \int_{-\infty}^{+\infty} \exp[-i(\mathbf{k}\mathbf{v}t' - k_t t')]dt', \tag{5}$$

and we obtain for the map of votes

$$\begin{aligned} m(\mathbf{U}, \mathbf{x}, t) = \iiint\int_{-\infty}^{+\infty} & \exp[-i(\mathbf{k}\mathbf{v}t' - k_t t')] \\ & \times \exp[i(\mathbf{k}\mathbf{x} - k_t t)] \\ & \times \exp[-(k_t - \mathbf{U}\mathbf{k})^2/(\xi|\mathbf{k}|)^2]d\mathbf{k}dk_t dt'. \end{aligned} \tag{6}$$

For matters of simplification, we assume $\xi \to 0$, and hence

$$\begin{aligned} & m(\mathbf{U}, \mathbf{x}, t) \\ & \cong \operatorname{sign}[I(\mathbf{x}, t)] \iiint\int_{-\infty}^{+\infty} \exp[-i(\mathbf{k}\mathbf{v}t' - k_t t')] \\ & \quad \times \exp[i(\mathbf{k}\mathbf{x} - k_t t)]\delta(k_t - \mathbf{U}\mathbf{k})d\mathbf{k}dk_t dt' \qquad (7) \\ & = \operatorname{sign}[I(\mathbf{x}, t)] \iiint_{-\infty}^{+\infty} \exp[i(-\mathbf{k}\mathbf{v}t' + \mathbf{U}\mathbf{k}t' \\ & \quad + \mathbf{k}\mathbf{x} - \mathbf{U}\mathbf{k}t)]d\mathbf{k}dt' \qquad (8) \\ & = \operatorname{sign}[I(\mathbf{x}, t)] \iiint_{-\infty}^{+\infty} \exp[-i\mathbf{k}(\mathbf{v} - \mathbf{U})t'] \\ & \quad \times \exp[-i\mathbf{k}(\mathbf{U}t - \mathbf{x})]d\mathbf{k}dt' \qquad (9) \\ & = \operatorname{sign}[I(\mathbf{x}, t)]\delta(\mathbf{v} - \mathbf{U}) \\ & \quad \times \iint_{-\infty}^{+\infty} \exp[-i\mathbf{k}(\mathbf{U}t - \mathbf{x})]d\mathbf{k} \qquad (10) \\ & = \operatorname{sign}[I(\mathbf{x}, t)]\delta(\mathbf{v} - \mathbf{U})\delta(\mathbf{U}t - \mathbf{x}), \qquad (11) \end{aligned}$$

which peaks only if $\mathbf{U} = \mathbf{v}$ and $\mathbf{x} = \mathbf{U}t$.

2.3 Algorithmic Extensions

Confidence Values We measure the "confidence" $G(\mathbf{x}, t)$ of the velocity estimated at $\mathbf{x}$ by computing the correlation coefficient of the map of votes with a Gaussian centered at $V_e(\mathbf{x}, t)$

$$\begin{aligned} G(\mathbf{x}, t) = C\big\{ & \exp\big[-(\mathbf{U} - \mathbf{V_e}(\mathbf{x}, t))^2/\sigma^2\big], \\ & m(\mathbf{U}, \mathbf{x}, t)\big\}, \end{aligned} \tag{12}$$

where σ is a parameter determining the width of the Gaussian. The function G provides a confidence measure for each estimated local velocity value. For example, if the velocity signal is too weak, no peak will emerge from the map of votes, and Eq. 4 will return a wrong velocity estimate. In this case, the confidence will be small and the velocity estimate can be discarded. We use a parameter τ to threshold the confidence function.

Regularization Many image sequences have largely smooth optic-flow fields, i.e. the image velocity changes continuously with space and time. In these cases, the performance of the optic-flow algorithm can be improved by a smoothing operation. In our algorithm, a regularizing smoothing step is introduced best before reading-out the velocity from the map of votes. We convolve the map of votes with a Gaussian function, giving

$$\tilde{m}(\mathbf{x}, t, \mathbf{U}) = m(\mathbf{x}, t, \mathbf{U}) * \exp\left(-\mathbf{x}^2/\alpha^2 - t^2/\beta^2\right), \tag{13}$$

where α and β are smoothing parameters. By performing this smoothing operation, the map of votes of neighboring points are combined. If the velocities of neighboring points are similar, the velocity map will improve. However, at motion boundaries, smoothing will introduce errors since signals belonging to different objects are falsely combined. To decrease the smoothing effects at the motion boundaries, a more elaborate smoothing method would have to be employed, i.e., global optimization with discontinuity-preserving constraints.

Preprocessing If an image point is part of a large homogeneous image area, the low spatial frequencies are usually static even if the respective object is moving. Gratings with a low spatial frequency need to be sampled over a longer time for a slow moving object in order to find their correct temporal frequency compared to a situation where the object is moving fast. If the sequence is short and motions are slow, we thus face a temporal aperture problem and it is advantageous to remove the affected Fourier components. Accordingly we preprocess the image sequence with a spatiotemporal high-pass Butterworth filter

$$\Psi_f = 1/[1 + \tau_f/(k_x^2 + k_y^2 + k_t^2)], \tag{14}$$

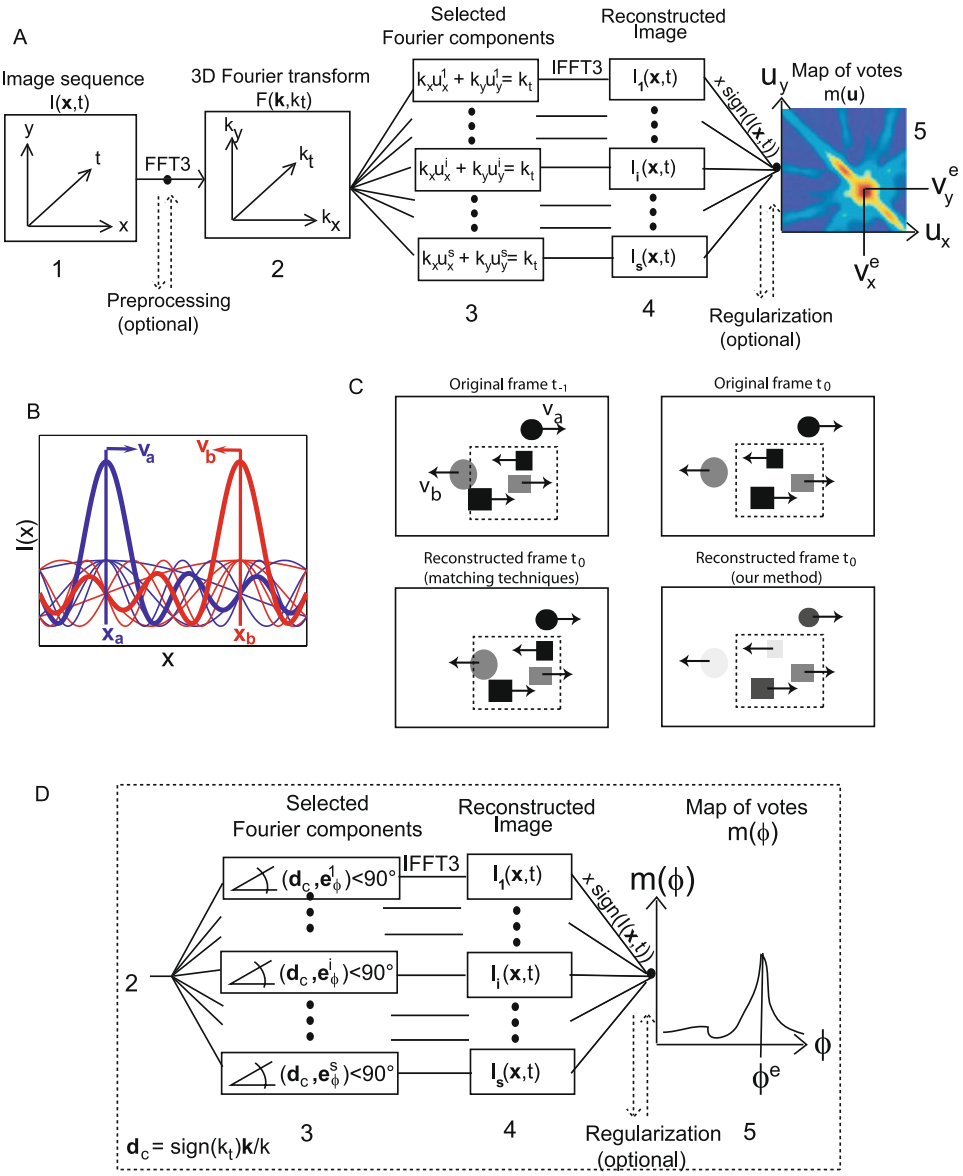

Fig. 1 (Color online) Algorithmic framework. **A** Algorithm for local-velocity computation. The fast Fourier transform (FFT3) of the image sequence is computed (step 1–2). The image sequence is preprocessed (optional) with a high-pass frequency filter. According to the motion-constraint equations, Fourier components are selected for a set of predefined test velocities $(\mathbf{u}_1, \ldots, \mathbf{u}_i, \ldots, \mathbf{u}_1)$ (step 3). For each velocity, the filtered image is reconstructed in real space (step 4), multiplied with the local sign of the image intensity, and in step 5 the resulting values for different test velocities are compared locally to find the velocity for which the value is maximal, as a consequence of constructive interference (see **B**). A regularizing smoothing operation may be employed to obtain dense flow fields. **B** Constructive interference: Spatiotemporal frequency of global Fourier components connect with local image velocity through the superposition principle. The Fourier components of two objects (*blue* and *red lines*), moving with v_a and v_b, superpose constructively (*bold blue* and *red lines*) at the instantaneous location of the objects, x_a and x_b, respectively. **C** Relation to region-based matching. Two consecutive frames of a motion sequence are shown (*upper left* and *right panel*). Shifting frame t_{-1} by test velocity $\mathbf{U} = \mathbf{v}_a$ results in the reconstructed frame t_0 shown in the lower, left panel, as performed in region-based matching. Applying the motion-constraint equation in global Fourier space with $\mathbf{U} = \mathbf{v}_a$ and returning to real space returns the reconstructed frame at t_0 shown in the *lower*, *right panel*. Objects moving with $\mathbf{v} \neq \mathbf{U}$ have faded in intensity, but appear at their original position. **D** Algorithm for local-motion-direction computation. Other motion features can be computed directly by modifying the frequency filters in step 3 of A. Fourier components are selected using a set of predefined motion directions $(\mathbf{e}^1_\phi, \ldots, \mathbf{e}^i_\phi, \ldots, \mathbf{e}^s_\phi)$, (step 3). The resulting map of votes (step 5) allows the detection of local motion direction

Table 1 Parameter choices

Sequence	No Regularization				With Regularization				
	ξ	σ	τ_f	Step size	ξ	τ_f	α	β	Step size
Trans. square	0.3	0.6	x	0.1	x	x	x	x	x
Lines	0.3	x	x	0.1/1	x	x	x	x	x
Transp. square	0.3	0.6	x	0.1	x	x	x	x	x
Trans. tree	0.3	0.6	0.1	0.1	0.6	0.2	15	3	0.05
Div. tree	0.3	0.6	0.1	0.1	0.6	0.2	15	3	0.05
Yosemite	0.3	0.6	0.2	0.1	0.6	0.2	15	3	0.05
Rubber whale	0.3	0.6	0.2	0.1	0.6	0.2	10	1	0.1
Hydrangea	0.3	0.6	0.2	0.1	0.6	0.2	10	1	0.2
SRI trees	0.3	0.6	0.05	0.1	0.6	0.2	5	1	0.2
Rubic cube	0.3	0.6	0.05	0.1	x	x	x	x	x
Hamburg taxi	0.3	0.6	0.05	0.1	0.6	0.2	10	1	0.2
Durlacher Tor	x	x	x	x	1	0.2	15	1	1
"Direction Fields"	x	x	x	x	1	0.2	15	1	30

where τ_f is a threshold parameter. A sensitivity analysis revealed that a large τ_f is advantageous at least for image sequences with smooth flow fields.

It should be noted that many optic-flow algorithms intrinsically perform high-pass filtering by not including Gabor filters of very low spatiotemporal frequencies or by computing image gradients.

2.4 Parameter Choices

Parameter choices of the algorithm are summarized in Table 1. According to a sensitivity analysis (Fig. 3), we chose ξ to be between 0.3 and 0.6 pixels/frame. If high-pass filtering is included as a preprocessing step, values in the range of $\tau_f = 0.05$ to 0.2 are reasonable choices. We tend to use a larger τ_f for shorter sequences because the amount of static gratings is expected to be higher. For the algorithm with smoothing (which we used for quantitative comparisons) we used fixed parameters $\tau_f = 0.2$ and $\xi = 0.6$ pixels/frame.

The parameter σ for obtained the confidence map should be chosen dependent on ξ, since the peak in the map of votes broadens with increasing ξ. Here, we chose $\sigma = 2\xi = 0.6$ pixels/frame.

3 Results

We first demonstrate the basic properties of the algorithm using three artificial test sequences (Sect. 3.1), and investigate its sensitivity to system parameters (Sect. 3.2). Then the algorithm both with and without smoothing is applied to several real image sequences and the error is evaluated for those sequences for which ground truth is available (Sect. 3.3). We further provide a qualitative discussion on aperture effects and their effect on regularization (Sect. 3.4). We then systematically study the effect of certain image properties, i.e., pattern speed, noise, and motion jitter, on the performance of the algorithm and in this context compare our results to region-based matching approaches (Anandan 1987; Singh 1990) (Sect. 3.5). In this context, the effect of noise is studied as well. Finally, we show that the proposed framework allows computing other motion features, e.g., motion-direction fields, as well (Sect. 3.6), which is an important characteristic of the approach. The absolute (endpoint) error and the angular error are defined as in Barron et al. (1994). Parameter choices are summarized in Table 1.

Computations were performed using MATLAB on a 1.73 GHz Intel Core Duo Processor. For a velocity range of -2 to 2 pixels/s and a step size of 0.1 pixels/frame, the program required 187 s to compute the optic-flow fields of an image sequences containing 20 frames of size 100×100 pixels, corresponding to an approximate computation time of 9 s/frame. However, the run time scales with a factor proportional to the ratio of the step sizes squared, i.e., for a step size of 1 pixels/frame we would obtain an approximate computation time of 0.09 s/frame.

3.1 Verification of the Core Algorithm and Characteristic Properties

In this section, we verify the core algorithm and its applicability to the motion problem. Characteristic properties of the method are elaborated.

We illustrate the algorithm first on a sequence which contains a square, 10 pixels wide, which moves on a white background with a velocity of $\mathbf{v} = [1, 1]$ pixels/frame. The translating-square sequence contains 24 frames, which is about the number of frames movies display within a second (Fig. 2A). The global Fourier transform is taken over the whole sequence, as explained in Sect. 2, and the weights are computed for each point of frame 12 of the sequence.

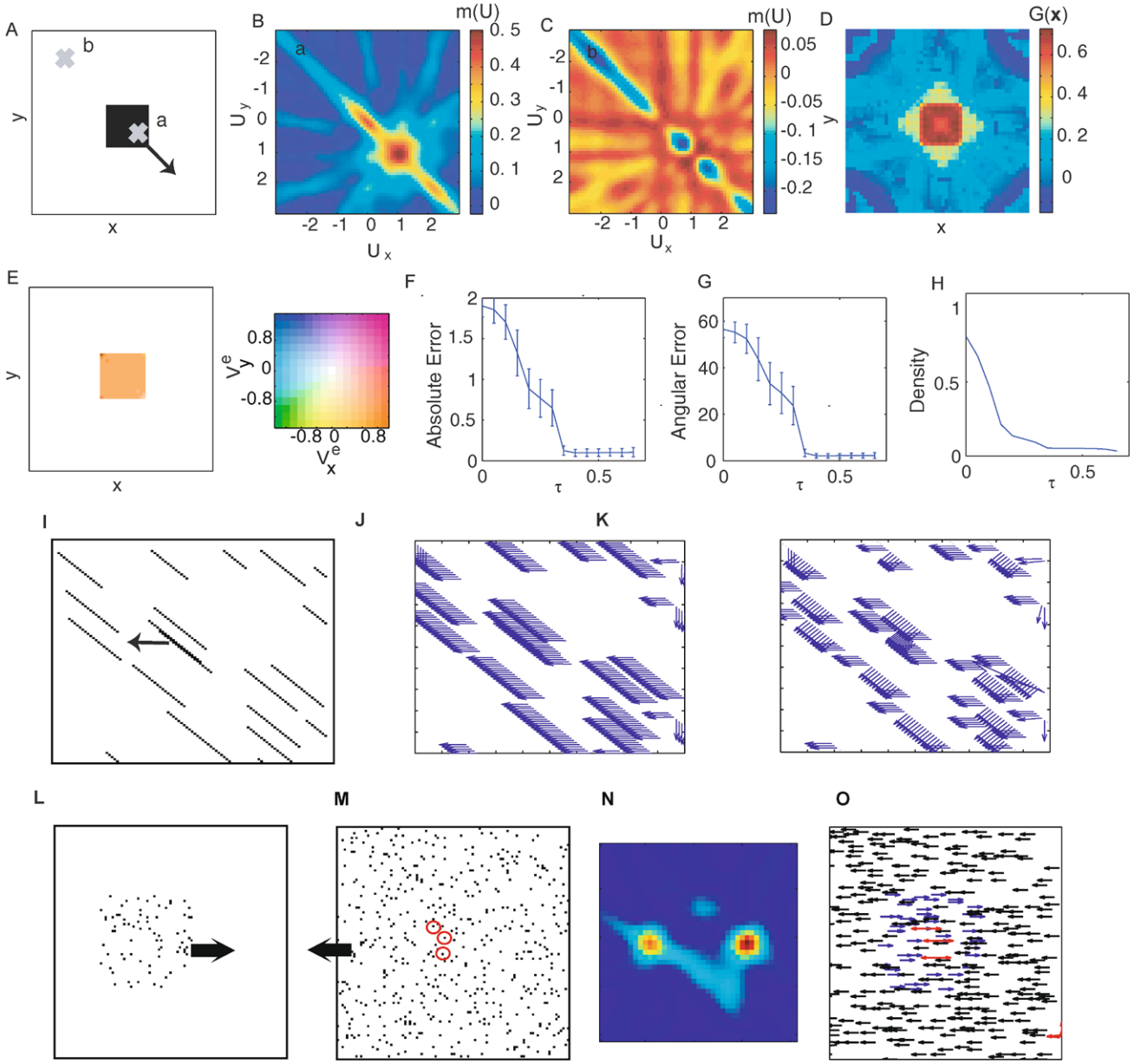

Fig. 2 (Color online) Results for artificial motion sequences (without regularization). **A** Schematic of translating square. **B** Map of votes for a point of the square, showing a clear peak, and **C** for point b in the homogeneous surround. **D** The confidence $G(\mathbf{x})$ as a function of position. **E** Color-coded estimated velocities and color code. **F–H** Absolute error, angular error, and density as a function of the threshold τ. **I** Schematic of oriented lines. **J** Computed optic-flow field (our approach). **K** Computed optic-flow field Lucas and Kanade for a window of size 5×5 pixels. **L–M** Fore- and background of a transparent motion. Positions at which two motion coincide are marked red. **N** Co-occurrence of two characteristic peaks in the map of votes at an image point containing two motions due to transparency. **O** Estimated optic-flow field. Transparent motions have been detected by identifying the characteristic peaks in the map of votes (respective velocity vectors are marked *red*)

We compute the map of votes for velocities $\mathbf{U}$, where the component velocities in x and y direction are ranging from -3 to 3 pixels/frame in steps of 0.1 pixels/frames. The map of votes for an image point belonging to the square (labeled a) is presented in Fig. 2B. A clearly distinguishable peak is visible in the map of votes, with a maximum positioned at $\mathbf{U} = [1, 1]$ pixels/frame.

In contrast, point b in the surround of the square, which is a large homogeneous image region, returns a flat map of votes, and no clear peak can be distinguished (Fig. 2C). This is reflected in the confidence $G(\mathbf{x})$ of the velocity estimate, presented in Fig. 2D. All points on the square have a high confidence, while points in the homogeneous surround have not. Thresholding $G(\mathbf{x})$ with $\tau = 0.4$ returns only velocity

Table 2 Average angular error E_a of our algorithm without (DW) and with smoothing regularization (DW, r). The densities of the respective flow fields are given in round brackets. The results are compared with region-based matching using our implementation (RBM, 1) (here a similar window size w as used for (DW, r) was chosen unless indicated otherwise) and Anandan (RBM, 2) (Barron et al. 1994), Lucas and Kanade (LK) (Lucas and Kanade 1981), and Gautama and van Hulle (GH) (Gautama and Van Hulle 2003)

Sequence	E_a^{DW}	$E_a^{DW,r}$	$E_a^{RBM,1}$	$E_a^{RBM,2}$	E_a^{LK}	E_a^{GH}
Lines	8.32/3.1(0.03)	x	$8.3_{w=14}/30_{w=5}(0.03)$	x	$18.2_{w=5}(0.03)$	x
Trans. tree	3.5(0.1)	0.52(1)	1.33(1)	4.54(1)[a]	0.66(0.4)[a]	2.67(0.68)[a]
Div. tree	4.37(0.044)	3.82(1)	13.99(1)	7.64(1)[a]	1.94(0.48)[a]	4.07(0.77)[a]
Yosemite	6.29(0.018)	3.75[1]/11.46(1)	$16.76/10.96_{w=14}(1)$	15.84[a]	6.41[b,1]/4.1(0.35)[a]	x/4.4(0.35)[a]
Rubber whale	5.49(0.012)	9.8(1)	17.76(1)	x	15.1(1)	10.14(0.6)
Hydrangea	12.52(0.014)	9.3(1)	12.80(1)	x	14.3(1)	6.86(0.37)

[a]Values taken from Barron et al. (1994), Gautama and Van Hulle (2003)

[b]Values taken from the Middlebury flow evaluation (LK with pyramids)

Otherwise parameters $N_{min} = 7$ and $\tau_l = 0.2$ were used for GH. For LK and RBM2, a window of 5×5 pixels was used if not indicated otherwise

[1]Clouds in Yosemite were excluded

Cases for which no values were available are marked with x

estimates which have a high confidence. The color-coded estimated velocities are plotted together with the color coding in Fig. 2E. The algorithm returns estimates for points lying on the square. Remarkably, the estimated velocities of points belonging to the edge of the square are not afflicted with the measurement-window-induced aperture problem, since no windowing is performed in real space and our method can thus exploit motion information contained in the whole image sequence. However, velocity estimates for points in the homogeneous surround cannot be given. This merely demonstrates the ambiguity of the vision problem (correspondence problem), and is not a insufficiency of the algorithm, since the homogeneous background contains all possible velocities.

To quantify the performance of the algorithm, we compute the absolute error and the angular error as a function of τ, together with the standard error (Fig. 2F–G). In Fig. 2H the density of the sequence is plotted as function of τ. The error measures decrease with increasing threshold and settle at low values close to the resolution of the map of votes of 0.1 pixels/frame.

To further demonstrate the behavior of our algorithm at edges, we designed an artificial image sequence consisting of oriented lines moving in negative x-direction with a speed of 1 pixels/frame (Fig. 2I). The sequence contains 30 frames. We plot the optic-flow field for the oriented lines computed with our algorithm (Fig. 2J) and the one of Lucas and Kanade with window size 5×5 pixels (Fig. 2K).[2] We observe that aperture effects are reduced in our method. We further computed the mean angular error only for the lines for our approach, Lucas and Kanade (1981), a phase-based approach (Gautama and Van Hulle 2003) (with $N_{min} = 7$ and $\tau_l = 0$), and region-based matching (for implementation details of region-based matching see Sect. 3.5) (Singh 1990; Anandan 1987). Results are presented in Table 2.

Finally, we apply the algorithm to an image sequence containing two moving random-dot patterns which are moving on top of each other under conditions of transparency (Fig. 2L–M). The random-dot patterns are moving with velocities $\mathbf{v} = (1, 0)$ pixels/frame and $\mathbf{v} = (-1, 0)$ pixels/frame, as depicted in Fig. 2L–M, respectively. In Fig. 2M, dots of the background which are overlapping with dots of the foreground are encircled by a red line. The respective map of votes shows two peaks, thus containing signals from both the pattern moving to the left and from the pattern moving to the right (Fig. 2N). Transparency does not eliminate the motion signal (as it would for some other methods), instead, co-occurrence of multiple peaks is observed in the map of votes, which can be used to detect transparent motion. We test for the presence of a second peak by removing the first detected peak, then locating the second peak by finding the maximum of the map of votes. We find a confidence value for transparent motion according to Eq. 13, except that the Gaussian contains two peaks. The second velocity estimate is then accepted if the confidence value for transparent motion is larger than the confidence value for single motion. The resulting optic-flow field is depicted in Fig. 2O. Confidence values were thresholded with $\tau = 0.75$. The velocities obtained at points of transparent motion are colored red. In the past, other methods have been proposed to detect multiple, transparent motions (Shizawa and Mase 1990; Aach et al. 2006;

[2]Note that in our implementation of Lucas and Kanade we used the gradient function of MATLAB to compute image gradients (more advanced techniques can be employed here) without image pyramids.

Mota et al. 2001). In Mota et al. (2001), a generalized structure tensor for multiple motions is derived from the output of linear filters, which allows the extraction of multiple motion vectors, which is shown analytically and also confirmed using an artificial test sequence. The detection of transparent motion is of relevance under semi-transparent conditions, which can occur for example in medical imaging (Mota et al. 2001).

In summary, we could show that correct velocity estimates could be obtained for extended edges, homogeneous and semi-transparent areas in certain situations where other local algorithms would have failed. It should be noted also that these results were obtained without applying smoothing, corresponding to a spatial window of only 1 pixel.

3.2 Sensitivity Analysis

First, we investigate the influence of the core parameter ξ on the computation of optic flow. We consider a Gaussian function moving with a constant velocity of 2 pixels/frame, as shown in Fig. 3A, left panel. In this example, the motion is one-dimensional. The map of votes at $(x,t) = (30,61)$ shows a clear peak at the position of the true velocity (dashed line) of the Gaussian for parameter values $\xi = 0.1$ pixels/frame, $\xi = 0.3$ pixels/frame, and $\xi = 0.6$ pixels/frame (2nd to 4th panel). We further observe a broadening of the peak with increasing ξ. When the Gaussian is accelerated with $a = 0.8$ pixels/frame^2 (Fig. 3B, left panel), the peak in the map of votes at $(x,t) = (50,25)$ deteriorates with decreasing ξ (Fig. 3B, 2nd to 4th panel), and the maximum of the peak is not exactly at the position of the true velocity (dashed line) of the image point. For large ξ, the maximum of the peak is located at the true velocity (dashed line) of the image point, but the peak itself is broadened.

Next, we investigate the performance of the algorithm in dependence of system parameters. We define the following performance measure $P = \sum_i |\rho_i / \overline{E}_{\phi,i}|$, where ρ_i is given in discrete steps of 0.01. This is done by writing the values of ρ_i and $\overline{E}_{\phi,i}$ in a histogram. While this measure is appropriate to quantify and compare performances of our method, it is not well suited to compare performances across methods or image sequences, because the density-error curves take different shapes or may be cut-off, thus affecting the result in a non-trivial manner. Thus, only relative performance are of significance here. In Fig. 3C, the gray-value coded performances for the translating-tree sequences are shown as function of ξ and α (left panel), ξ and the test-velocity step size (middle panel), and τ_f and the number of frames used (right panel). A lighter gray value indicates a better performance. The same experiments were performed for the diverging-tree sequence (Fig. 3D). The translating- and diverging-tree sequences are well-known artificial sequences generated from a real image of a tree (Barron et al. 1994) (see Fig. 5B). In the translating-tree sequence, the camera moves to the right, while filming a picture of a tree, which is slanted with respect to the front-parallel plane. The velocities are parallel to the x-axis and range form 1.73 to 2.26 pixels/frame. In the diverging-tree sequence, the camera recedes from the tree image, causing the optic-flow field to diverge from the center of expansion which is located in the middle of the image. The speed ranges from 1.29 to 1.86 pixels/frame.

The results show that the algorithm performs best for large values of the smoothing parameter α, however, this is not surprising because the ground truth of the translating- and diverging-tree sequence are smooth global motion patterns. We further find that ξ between 0.3 and 1 pixel/frame to be a reasonably good value for the image sequences considered. In general, we observe that a small step size is best, except for the translating-tree sequence for which a step size of 1 pixel/frame is preferable, but this is most likely due to the particular velocity distribution of the ground truth. We further conclude from the sensitivity analysis that the performance of the algorithm is best for a large number of frames. This is understandable since the 3D Fourier transform will deliver only a coarse temporal frequency range for short image sequences. The result further demonstrates that the algorithm utilizes motion information contained in the entire sequence. We also observe that a large parameter τ_f improves the results.

Parameter choices (if parameter is not varied) have been $\xi = 0.6$ pixels/frame, $\alpha = 7$ pixels, $\beta = 1$ frame, and $\tau_f = 0.2$.

3.3 Real Image Sequences

We apply the algorithm to several real images sequences. First we show results without smoothing to demonstrate that with our method it is indeed possible to obtain results under conditions for which many local methods would fail, i.e. a spatial window of 1 pixel.

In the SRI sequence, a camera moves parallel to the ground plane along the x-axis in front of several trees. The velocities are as large as two pixels/frame. The sequence contains 20 frames. The color-coded velocities in x direction of frame 10 for $\tau = 0.3$ and $\tau_f = 0.05$ are shown in Fig. 4A, right panel. Image points which are below threshold are set to zero. The tree in the foreground has a speed of about 2 pixels/frame and be distinguished from the background despite the poor resolution, the amount of occlusion, and the low contrast of the image sequence.

In the Rubic-cube sequence, a Rubic cube is rotating counterclockwise on a turntable. The rotation induces velocities of the cube between 0.2 and 0.5 pixels/frame. The velocities on the turntable (Fig. 4B, left panel) range from 1.2 to 1.4 pixels/frame. This sequence contains 20 frames.

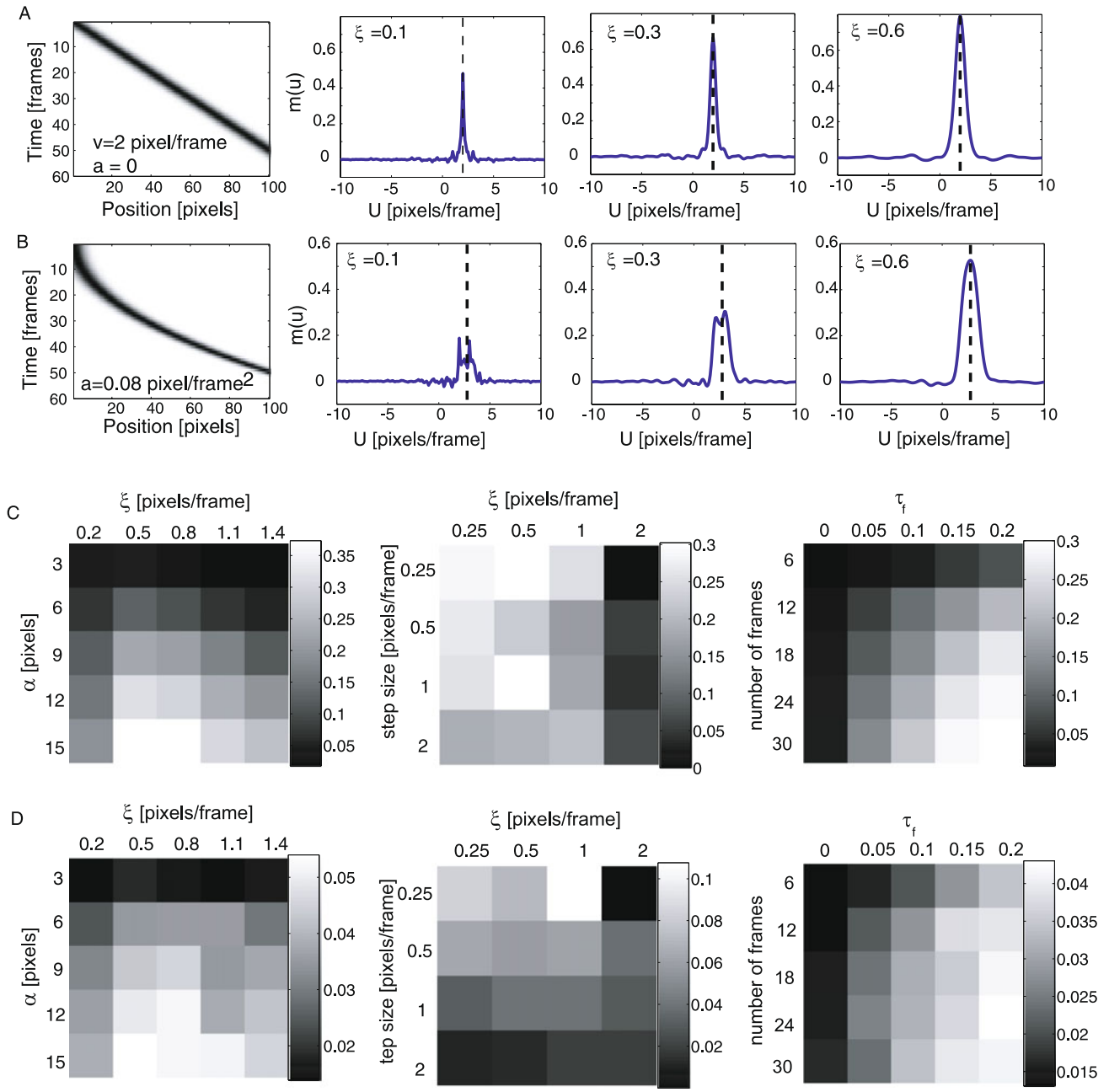

Fig. 3 Sensitivity analysis. **A** The x-t plot of a Gaussian function moving with a constant velocity of 2 pixel/frame is shown. The corresponding maps of votes for $\xi = 0.1$, $\xi = 0.3$, and $\xi = 0.6$ pixels/frame show a broadening of the peak with increasing ξ. **B** The x-t plot of a Gaussian function moving with a constant acceleration of 0.08 pixels/frame^2 is shown. The corresponding map of votes for $\xi = 0.1$, $\xi = 0.3$, and $\xi = 0.6$ pixels/frame. The peak deteriorates with decreasing ξ. **C–D** The performance of the algorithm is measured as a function of several parameters for the translating- (**C**) and diverging-tree sequence (**D**). The performance values have been gray-value coded, where a *lighter tone* indicates a better performance than a *darker tone*. For further information, see text

The optic-flow field at frame 10 for $\tau = 0.3$ and $\tau_f = 0.05$ is shown in Fig. 4B, right panel. The optic-flow field is scaled by a factor 2 for reasons of better display. This demonstrates that our method is applicable to rotational motion as well.

In the Hamburg taxi sequence, a street scene is shown with four moving objects: (i) a taxi turning the corner (ii) a car in the lower left, driving from the left to the right (iii) a van in the lower right driving from the right to the left, and (iv) a person in the upper left walking downwards and to the left. Image speeds of the four moving objects are approximately 1.0, 3.0, 3.0, and 0.3 pixels/frame, respectively. A snapshot shown in Fig. 4C. The sequence contains

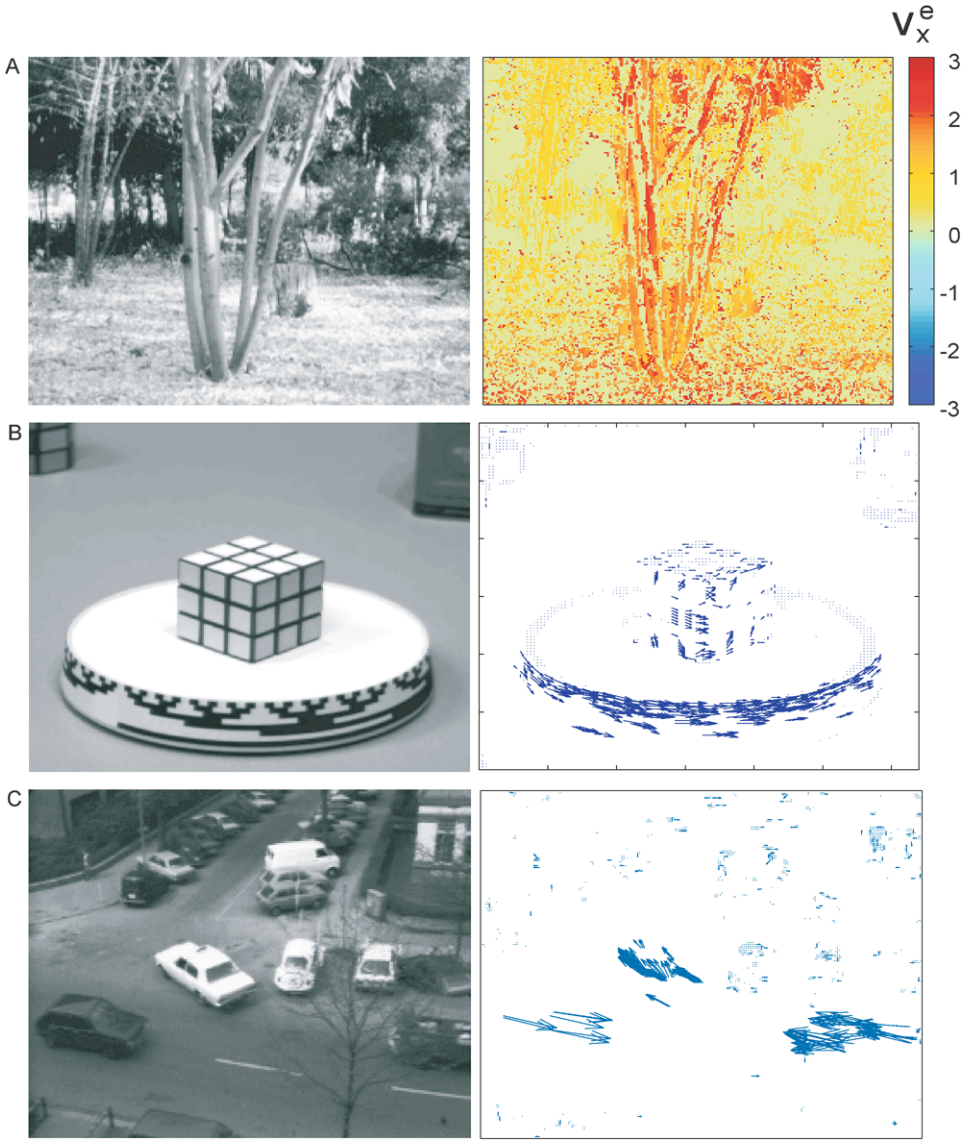

Fig. 4 Estimated optic-flow field for real-image sequence without ground truth (without regularization). **A** SRI-tree sequence. Frame 10 of the original sequence (*left panel*). Estimated optic-flow field at frame 10 for $\tau = 0.3$ and $\tau_f = 0.05$ (*right panel*). **B** Rubic-cube sequence. Frame 10 of the original sequence (*left panel*). Estimated optic-flow field at frame 10 for $\tau = 0.3$ and $\tau_f = 0.05$ (*right panel*). **C** Hamburg taxi sequence: Frame 10 of the original sequence (*left panel*). Estimated optic-flow field at frame 10 for $\tau = 0.55$ and $\tau_f = 0.05$ (*right panel*)

20 frames. The optic-flow field at frame 10 for $\tau = 0.55$ and $\tau_f = 0.05$ is shown in Fig. 4C, left panel. The optic-flow field is scaled by a factor 2 for reasons of better display. By means of manual segmentation, we computed the average velocity for each moving object: (i) $\bar{\mathbf{V}}_e = (-0.7, -0.7)$, (ii) $\bar{\mathbf{V}}_e = (2.7, 0.5)$, (iii) $\bar{\mathbf{V}}_e = (-2.6, -0.3)$, and (iv) $\bar{\mathbf{V}}_e = (-0.3, 0)$ pixels/frame.

We included the smoothing operation as described in Sect. 2.3 and applied the modified algorithm for several real image sequences. In Fig. 5A, left panel, the estimated velocity in x-direction for the SRI-tree sequence is shown. The parameter choices are $\tau_f = 0.2$, $\xi = 0.6$ pixels/frame, $\alpha = 5$ pixels, and $\beta = 1$ frame. Even though the optic-flow fields captured the main velocity pattern of the sequence, undesirable but expected smoothing effects are visible at the motion boundaries (Fig. 5A).

Next, we applied the algorithm with smoothing to the translating- and diverging-tree sequence with parameters $\xi = 0.6$, $\tau_f = 0.2$ pixels/frame, $\alpha = 15$ pixels, and $\beta = 3$ frames (Fig. 5B). The algorithm returns flow fields with 100% density and angular errors of 0.52 deg for the translating-tree sequence and 3.82 deg for the diverging-tree sequence.

For the taxi sequence, our algorithm with parameters $\tau_f = 0.2$, $\xi = 0.6$ pixels/frame, $\alpha = 10$ pixels, and $\beta = 1$ frame returns a dense optic-flow field in which the moving objects are clearly visible (Fig. 5C, left panel). The velocity estimates are close to the true velocities of the objects. However, the objects appear larger than they are due to regularization.

We also computed optic-flow fields for rubber whale with $\tau_f = 0.2$, $\xi = 0.6$ pixels/frame, $\alpha = 10$ pixels and $\beta = 1$ frame, and, alternatively, $\alpha = 10$ pixels and $\beta = 5$ frame

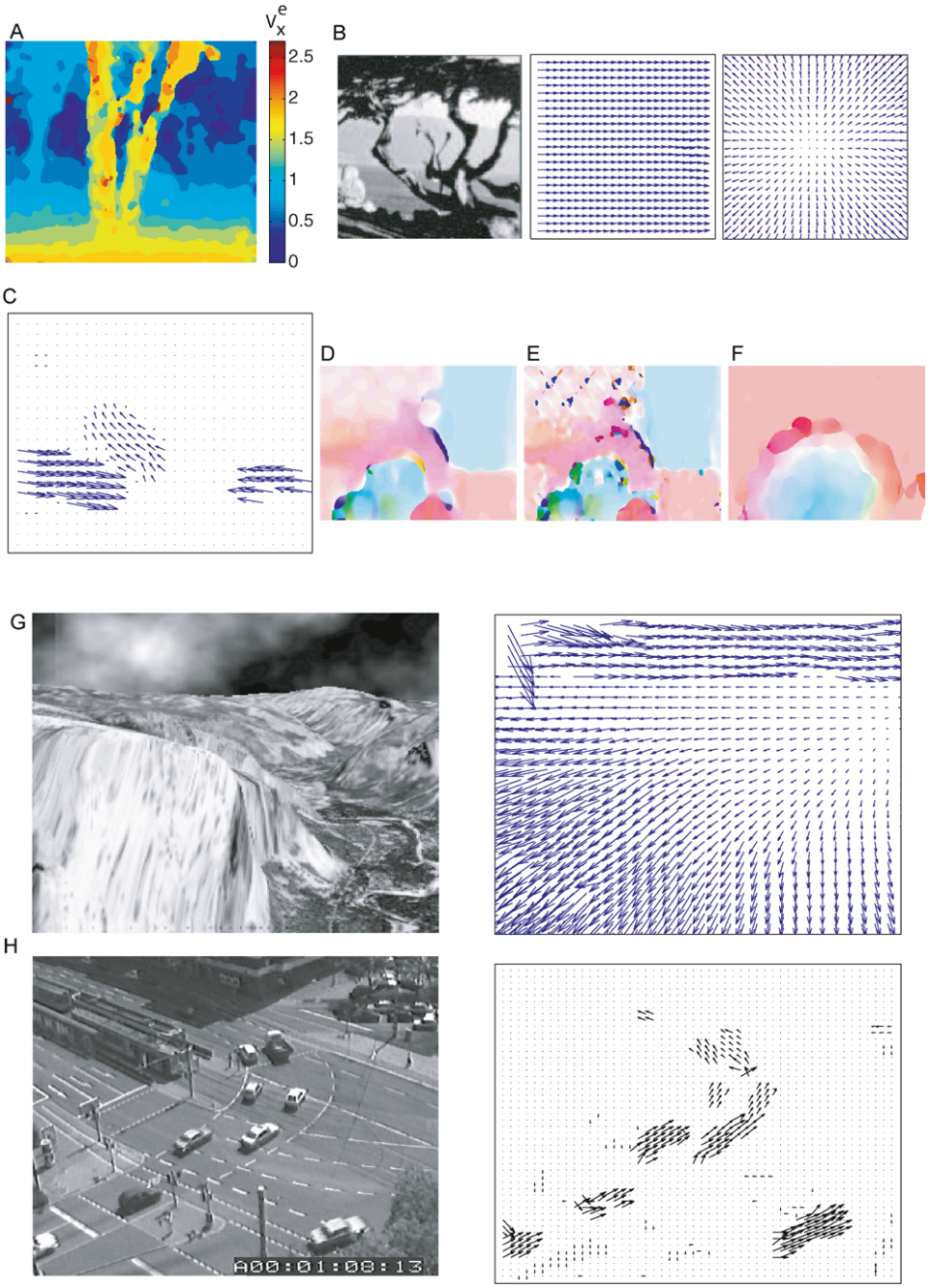

Fig. 5 Estimated optic-flow fields for several real image sequences (with regularization). **A** SRI-tree sequence. **B** Translating/diverging tree. **C** Hamburg taxi. **D** Rubber whale ($\alpha = 10$ pixels). **E** Rubber whale ($\alpha = 5$ pixels) **F** Hydrangea. **G** Yosemite snapshot (*left panel*). Estimated optic flow (*right panel*). **H** Durlacher Tor snapshot (*left panel*). Estimated optic flow (*right panel*)

(Fig. 5D–E). The results for hydrangea with $\alpha = 10$ pixels and $\beta = 1$ frame are shown in Fig. 5F. Images have been taken from the Middlebury database. We obtained an angular error of 9.8 deg and 9.3 deg at 100% density for rubber whale and hydrangea, respectively. We used here a downsized version (factor 2) of rubber whale and hydrangea. However, since the angular error is used for quantitative comparisons, this choice should have only minor consequences on the results.

For the Yosemite sequence (Fig. 5G, right panel) with parameters $\xi = 0.6$ pixels/frame, $\alpha = 15$ pixels, and $\beta =$ 3 frames, the algorithm returns flow fields with 100% density and angular errors of 0.52 deg for the translating-tree sequence and 3.82 deg for the diverging-tree sequence (Fig. 5G, left panel). The algorithm achieves an angular error of 3.75 everywhere except for the area of the clouds. In the cloud area, the angular error is 35.25 deg. However, this error has been computed under the assumption that the clouds are moving with a constant velocity of 1 pixel/frame. It should be noted that this value only approximates the true motion of the clouds, which are undergoing Brownian mo-

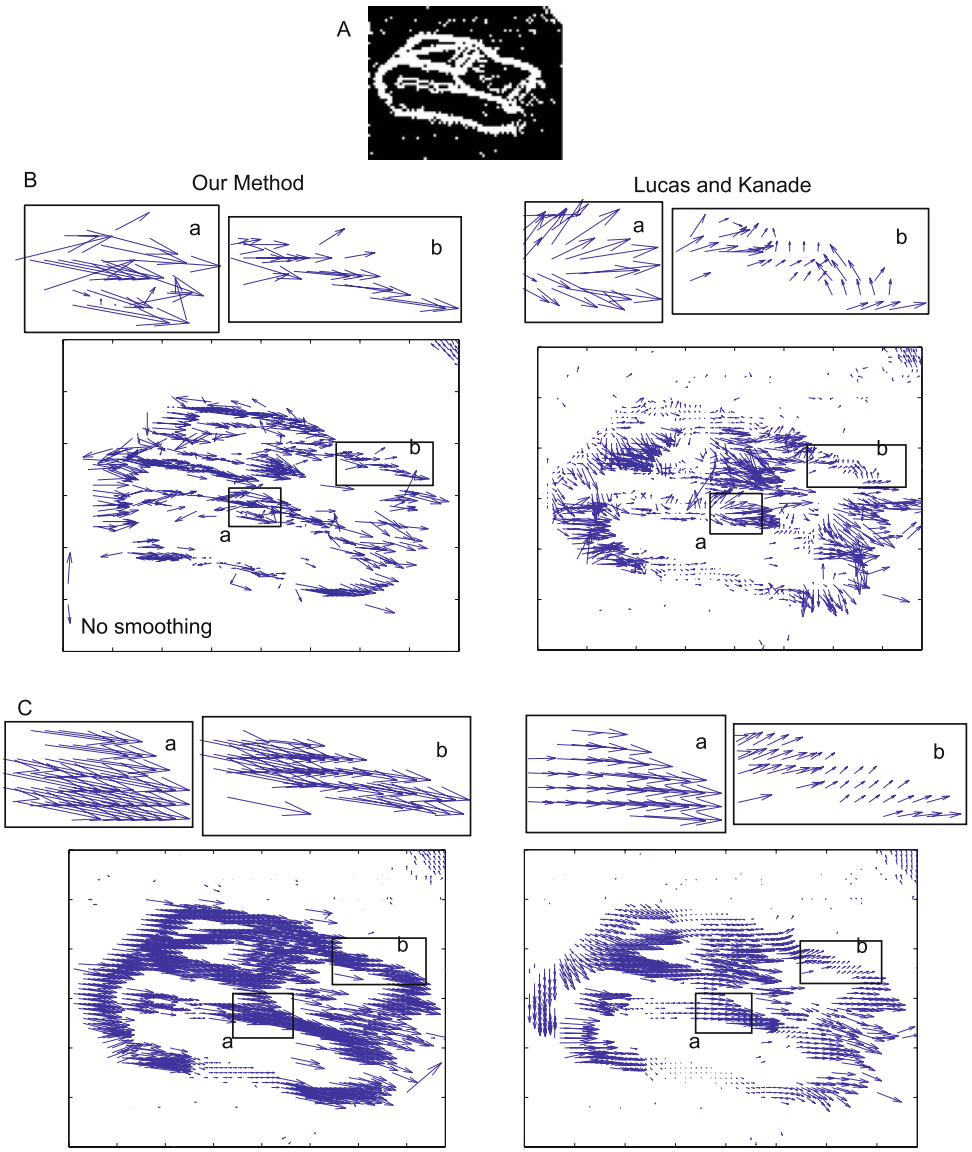

Fig. 6 Aperture effects and their effect on regularization. **A** Thresholded absolute image gradient of a frame of the Hamburg taxi sequence, showing just the car in the left lower corner. In the following, only velocity estimates for pixels above threshold are displayed to highlight computed flow features at edges. **B** Results of our method without regularization (*left panel*) and the one of Lucas and Kanade (1981) with a window size of 3×3 pixels (*right panel*), ensuring equal locality. Aperture effects are clearly visible for the method of Lucas and Kanade (1981). **C** Results of our method with smoothing only along the spatial direction ($\alpha = 5$ pixels) (*left panel*) and the one of Lucas and Kanade (1981) using a similar window size of $w = 10 \times 10$ pixels (*right panel*)

tion and are changing shape. Hence, errors reported for the cloud region do not have much significance.

We also applied the algorithm with smoothing to the Durlacher-Tor sequence (KOGS/IAKS Universität Karlsruhe), showing a traffic intersection scene recorded at the Durlacher-Tor-Platz in Karlsruhe by a stationary camera (Fig. 5H, left panel). For this sequence, we used parameters $\tau_f = 0.1$, $\xi = 1$ pixels/frame, $\alpha = 15$ pixels, and $\beta = 1$ frame, and a large step size of 1 pixel/frame. The various moving objects can be clearly identified in the computed optic-flow field (Fig. 5H, right panel).

Quantitative results are summarized in Table 2 together with results obtained with other local approaches. The comparison shows that our approach delivers similar and in several cases even better results than other local approaches such as region-based matching, differential methods, and phase-based methods.

3.4 The Aperture Problem and Its Effect on Regularization

We investigate qualitatively differences between our method and another representative local method, for which we chose the algorithm of Lucas and Kanade. We compare our method without smoothing with Lukas and Kanade for a window size of 3×3 pixels, because only for this value comparable locality with our method is achieved, but note that regularization is introduced for window sizes larger than one pixel. This comparison demonstrates that the measurement-window-induced aperture problem is reduced in our method. For this purpose, we display previously computed velocity estimates for the Hamburg taxi sequence only at image ar-

eas having a large image gradient (see Fig. 6A). The selected velocity vectors obtained by our method with $\tau = 0$ and the method of Lucas and Kanade (1981) for a window size of 3×3 pixels are shown in Fig. 6B, left and right panel, respectively, for the car moving from the left to the right. Measurement-induced aperture effects are clearly visible for the method of Lucas and Kanade (1981), whereas in our case, local velocity estimates, capturing the true motion of the car, could be assigned to the elongated edges of the car. These differences are clearly visible in the magnifications.

Increasing the window size of the Lucas and Kanade algorithm to 10×10 pixels does partly remove the aperture problem (see Fig. 6C, right panel), however, as a consequence motion boundaries will be blurred. In addition, as soon as there is a too dramatic lack of information (mag. b from panel C) the correct flow cannot be recovered by the Lucas and Kanade method. For our method we used $\alpha = 5$ pixels and no smoothing along the temporal dimension. To allow a fair comparison in terms of regularization, we used a Gaussian window for Lucas and Kanade of the same size and shape as in our method.

Using a larger window size in the Lucas and Kanade algorithm acts regularizing. Clearly there are nowadays much more powerful methods for flow-regularization and global optimization existing (Weickert and Schnörr 2001; Bruhn et al. 2005; Papenberg et al. 2006; Felzenszwalb and Huttenlocher 2006), but the fundamental problem remains: As soon as flow information at edges becomes too sparse, regularization will become more difficult. Here, we see one potential major advantage of our new algorithm as we are less affected by the measurement-window-induced aperture problem[3] having the consequence that in certain areas information can be extracted where other methods fail. The reason is that our method takes into account the whole image sequence during the motion analysis, and not only a small area of the image sequence.

3.5 Performance with Respect to Image-Intrinsic Properties and Comparison with Region-Based Matching

We investigate the effect of certain image properties, i.e., pattern speed, additive noise, and motion jitter, on the performance of our algorithm. The results are compared with those obtained for region-based matching. Region-based matching is in some sense related to our method and as such allows us to compare the "core" methods directly with each other.

Filtering for a certain test velocity $\mathbf{U}$ in global Fourier space and returning to real space using the principle of constructive interference yields an "reconstructed" image which can be directly compared with the original. Region-based matching also provides a "reconstruction" for a certain test velocity, which can be obtained by shifting all pixels of an adjacent frame uniformly. This is illustrated in Fig. 1C. In the left and right upper panel, two consecutive frame t_{-1} and t_0 of the image sequences are shown. For test velocity $\mathbf{U} = v_a$, region-based matching reconstructs frame t_0 by shifting all objects by v_a, as illustrated in the lower, left panel. In contrast, our method reconstructs the image at t_0 from the global Fourier components, after applying the motion-constraint equation in Fourier space for $\mathbf{U} = v_a$. The reconstructed image is shown in the lower right panel. All objects moving with v_a have been fully reconstructed, while the remaining ones have faded in intensity. In both cases, local velocity estimates are obtained by comparing the reconstructed with the original image for a set of test velocities.

We adjusted both methods such that only the core algorithms would differ, but all other additional techniques would be identical. For the regularization window of our algorithm, we use a rectangular spatial window of size 15×15 pixels and no regularization along the temporal dimension. For the region-based matching algorithm, an identical window is chosen. In both algorithms the image sequence is preprocessed with an identical high-pass spatiotemporal filter with $\tau_f = 0.2$ pixels. We note that the results overall of region-based matching are improved if preprocessing is applied.

Furthermore, both methods provide a map of votes, which allows utilizing the same velocity-read-out technique. In region-based matching, the sum-of-squared-differences (SSD) of pixel gray values of the reference image at t_0 shifted by a test velocity $\mathbf{U}$ with the adjacent images t_{-1} and t_{+1} is computed within a rectangular measurement window (Barron et al. 1994), providing a set of SSD values, which we subtract from the maximum SSD value of all test velocities. This returns a map of votes very similar to our method. Since region-based matching only allows integer pixel displacement, we also constrain our algorithm to integer velocities, and modify our velocity-read-out technique as follows. The maximum in the map of votes is determined and an average velocity is computed from the test velocities within an rectangular window of size 3×3 around the maximum. A confidence value is computed in a simplified way by calculating the correlation coefficient between the map of votes and a map that contains a single peak at the position of the maximum. For our algorithm we choose $\xi = 0.6$ pixels/frame and a total number of 24 frames.

[3] For the sake of completeness, we emphasize that aperture problems may also arise due to insufficient velocity information carried by the image sequence, which we identify as the correspondence problem. As an example, no unique velocity can be assigned to points belonging to an image consisting of a uniform surface. Such ambiguities are intrinsic to the data and can thus not be removed by applying a different measurement technique.

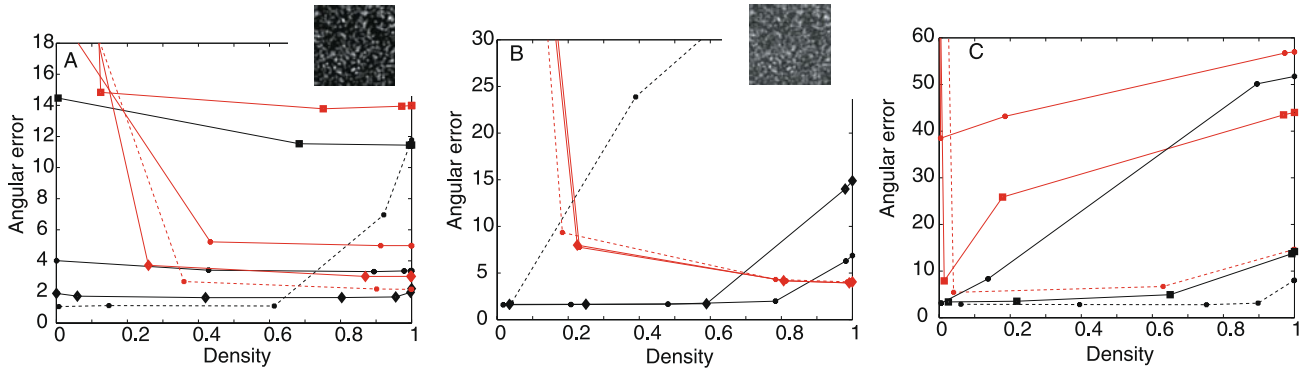

Fig. 7 Effect of image properties on performance in comparison with region-based matching. **A** Angular error for a translating random dot pattern moving at different image speeds. **B** Angular error for a translating random dot pattern with different levels of Gaussian noise added to the images. **C** Angular error for a translating random dot pattern with Gaussian motion jitter added to the dot trajectories. Results of our method are plotted in black, while the ones obtained for region-based matching are plotted in red. For further explanations, see text

First, we compare the performance of both methods with respect to image speed. In Fig. 7A, the angular-error/density curves are shown for a translating random Gaussian dot pattern (see inset Fig. 7A) moving at velocity $(v_x, v_y) = (0.5, 1)$ (solid line with square), $(v_x, v_y) = (2.5, 3)$ (solid line with circle), $(v_x, v_y) = (4.5, 5)$ (solid line with diamond), $(v_x, v_y) = (6.5, 7)$ pixels/frame (dashed line with circle). Except for the first case at high densities, our method clearly outperforms region-based matching. The image sequence contained 24 frames of size 200×200 pixels. For large velocities, our method breaks down at some point if not more frames are added. Otherwise a temporal aperture is created which prevents sufficient sampling along the temporal dimension of the Fourier transform.

Here and for the remaining simulations, we used a fixed step size of 1 pixel/frame and a fixed maximum absolute test velocity of 8 pixels/frame.

Now we add Gaussian distributed noise to the image sequence with different standard deviations of 0.025 (solid line with circle), 0.05 (solid line with diamond), 0.075 (dashed line with circle), given as percentage of the maximum intensity of the image. Here, we again observe that our method performs best at low densities, while at higher densities region-based matching shows better results (Fig. 7B). The velocity of the input pattern had been chosen as $(v_x, v_y) = (4.5, 5)$ pixels/frame.

Finally, we measure the angular error of both methods for different amounts of motion jitter. To each random dot, we added a Gaussian distributed random motion component to the translating motion, which caused each dot to perform a jittery movement along its trajectory. As ground truth, we assume the original translating movement without jitter. We add Gaussian-distributed jitter with a standard deviation of 1 (dashed line with circle), 1.5 (solid line with square), and 2 pixels/frame (solid line with circle), where the initial velocity of the input pattern had been chosen as $(v_x, v_y) = (3.5, 4)$ pixels/frame (see Fig. 7C).

As we can see in Fig. 7C, adding jitter drastically decreases performance of region based matching, while our method is highly robust to this kind of noise. This suggests that our method may be well applicable to tracking of particle images which often perform some additional jittery motion. Similar results are observed (not shown) if adding jitter uniformly to the whole frame, as it could be expected if the camera is vibrating slightly due to wind or other environmental factors.

3.6 Local Motion Direction

We demonstrate that the core of our method is different from previous approaches by modifying our computation scheme (Fig. 1A) such that local motion direction is computed instead local velocity. This only requires that the frequency filters for local velocity are replaced by frequency filters for local motion direction as follows. Each Fourier component votes with its assigned weight for all local motion directions $\mathbf{e}_\phi$ if the angle enclosed by the component motion direction vector $\mathbf{d}_c = \text{sign}(k_t)\mathbf{k}/k$ and the vector $\mathbf{e}_\phi$ is smaller than 90 deg, yielding a map of votes

$$m(\phi, \mathbf{x}, t) = \iiint_{-\infty}^{+\infty} w(\mathbf{x}, t, \mathbf{k}, k_t, \mathbf{x}, t) \times \theta(\angle(\mathbf{d}_c, \mathbf{e}_\phi), 90) dk_x dk_y dk_t, \quad (15)$$

where the parameter $\theta(a, b) = 1$ if $a < b$ and zero otherwise. As for the local velocity estimation, the position of the maximum of $m(\phi, \mathbf{x}, t)$ defines the estimated local motion direction at $(\mathbf{x}, t)$. Note that the computation of motion direction does not require an intermediate representation/computation of local velocity. The results of the algorithm with smoothing parameters $\alpha = 15$ and $\beta = 1$ and a

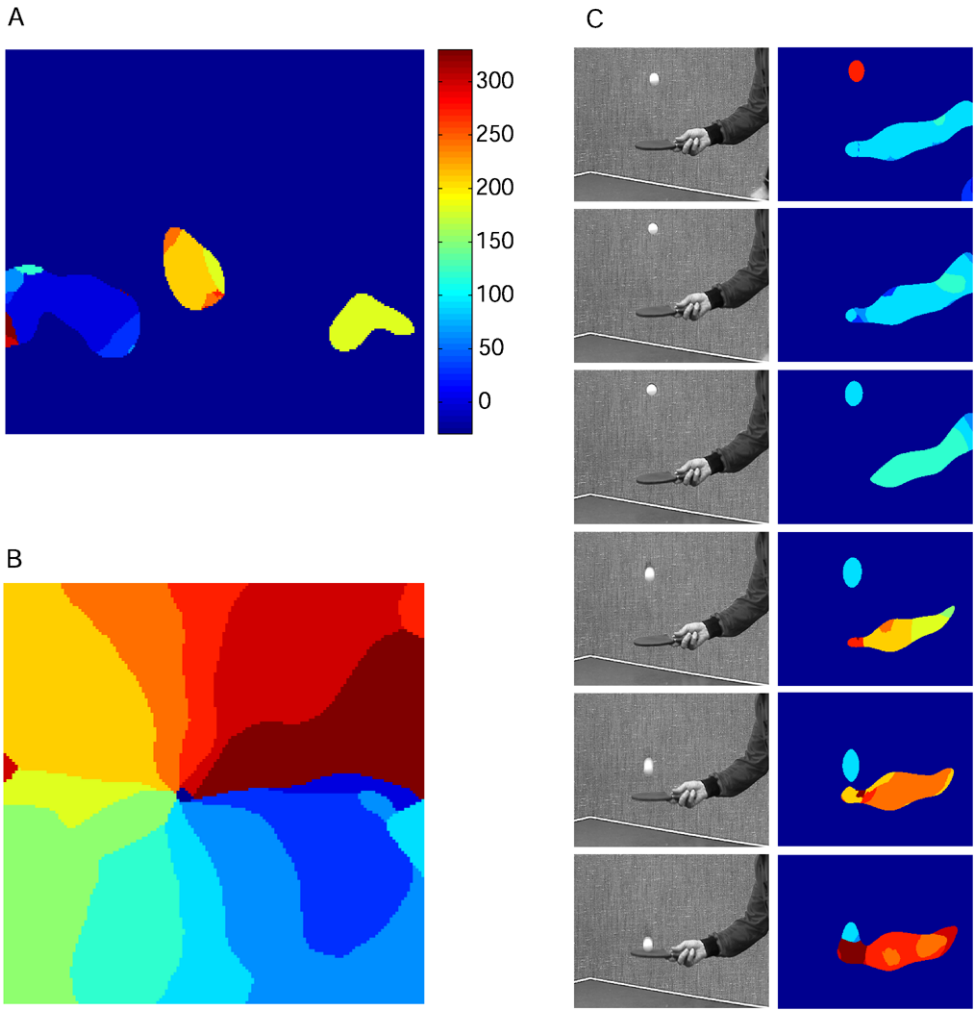

Fig. 8 Estimated motion-direction fields (with regularization). **A** Hamburg taxi with color code in degrees with respect to the positive x-axis. **B** Diverging tree. **C** Tennis. The fast moving ball could be detected by the method

step size of 30 deg for Hamburg Taxi, Diverging Tree, and Tennis (OSU/SAMPL database) are presented in Fig. 8A–C, respectively. The motion directions of the objects could be captured by the algorithm. Furthermore, the algorithm succeeded in capturing the fast motion of the ball in Tennis, which constitutes a major challenge for most algorithms (except matching techniques).

4 Discussion

We proposed a novel local algorithm (i.e. no global optimization is performed) which derives velocity estimates from the global Fourier components of the image sequence using the principle of constructive interference. The motion-constraint equation is formulated in Fourier space and corresponding spatiotemporal frequency filters are applied to select certain motion features. Then spatiotemporal position is reconstructed from the phases and amplitudes of the filtered Fourier components of the image sequence. The "reconstructed" image can then be directly compared with the original image for local velocity estimation, as illustrated in Fig. 1C.

We demonstrated the characteristic properties of the algorithm on several artificial image sequences. In particular, we could show that correct velocity estimates could be obtained for several examples at extended edges and in homogeneous areas for which other local algorithms would have failed. We also showed that the method is applicable to the detection of transparent motion. The sensitivity of the algorithm with respect to system parameters was analyzed and a suitable parameter range for further simulations was identified.

We further applied the algorithm to real image sequences. The basic motion structure of the scene could always be captured. We found low mean angular errors at low densities. Since the algorithm does not use a (regularizing) measurement window, the resulting local-velocity fields are very sparse, making a comparison with other methods difficult. We included a regularizing smoothing operation which allowed an easier comparison of our results with other local methods (see Table 2). For scenes with largely smooth flow

fields (translating/diverging tree, Yosemite) dense optic-flow fields with a low mean angular error could be obtained. For scenes containing many motion discontinuities (rubber whale, hydrangea), smoothing densified the flow fields, but the angular error did not drop significantly. The same effect could be observed for two other local methods. In summary, the regularized optic-flow algorithm delivered results competitive to other local methods and even outperformed them in terms of mean angular error in several cases. This improvement in performance may be caused by the different behavior of our method along extended edges compared with other local methods. We demonstrated on the example of the Hamburg taxi sequence that our method delivers qualitatively better results along an extended edge than the method of Lucas and Kanade (1981), where measurement-induced aperture problems become apparent, and quantitatively for an artificial image sequence of moving oriented lines. Importantly, smoothing cannot always recover these kinds of aperture problems, suggesting that similar effects might occur in global methods as well. Finally, we showed that our framework can also be applied to extract other local motion features, here motion direction, directly from the global Fourier components, which is an important characteristic of the method (in particular compared to matching approaches). The algorithm is applicable to image sequences containing motions with large displacements. For the tennis sequence, the motion of the fast moving ball could be captured and tracked from frame to frame (Fig. 8C).

We further investigated the performance of our algorithm with respect to various image properties, i.e., pattern speed, additive noise, and motion jitter, and in this context compared the algorithm with region-based matching. For this analysis, we assimilated both algorithms and all system parameters as much as possible to allow for a fair evaluation. As a consequence, we had to use a large step size of 1 pixel/frame also for our algorithm, even though our method also allows for sub-pixel step sizes and delivers better results for these. However, larger step sizes have the advantage that the run time of the algorithm decreases. Both our method and region-based matching have been shown to be applicable to fast moving stimuli, while our method performed on average better than region-based matching. Gaussian noise added to the image decreased both the performance of our algorithm and the region-based matching, however, the characteristics differ. For motion noise, our algorithm showed strictly better results than region-based matching.

Our method is computationally expensive, being the main disadvantage of the approach compared to, e.g., Lucas and Kanade. The speed of the algorithm is mainly determined by the speed of the 3D fast Fourier transform and the number of test velocities used. However the full 3D transform does not need to be computed for each frame, because results from previous computations can be reused. For an image sequence of size $100 \times 100 \times 20$ we estimate an approximate run time of 0.09 s/frame on an Intel Core Duo Processor with 1.73 GHz using a MATLAB implementation for a step size of 1 pixel/frame. Run times could potentially be improved by performing the Fourier transformation on a graphical processing unit or by using a multi-core architecture where the computations for different test velocities are executed in parallel.

Acknowledgements The work has received support by the German Ministry for Education and Research (BMBF) via the Bernstein Center for Computational Neuroscience (BCCN) Göttingen under Grant No. 01GQ0430 and the EU project Drivsco under contract number 016276-2. B.D. also acknowledges support from the Spanish Ministry for Science and Innovation via a Ramon y Cajal Fellowship.

References

Aach, T., Mota, C., Stuke, I., Muhlich, M., & Barth, E. (2006). Analysis of superimposed oriented patterns. *IEEE Transactions on Image Processing*, *15*(12), 3690–3700.

Adelson, E. H., & Bergen, J. R. (1985). Spatiotemporal energy models for the perception of motion. *Journal of Optical Society of America A*, *2*, 284–299.

Anandan, P. (1987). A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, *2*, 283–310.

Baker, S., Scharstein, D., Lewis, J. P., Roth, S., Black, M., & Szeliski, R. (2007). A database and evaluation methodology for optical flow. In *ICCV 2007*.

Barron, J. L., Fleet, D. J., Beauchemin, S., & Burkitt, T. (1994). Performance of optical flow techniques. *International Journal of Computer Vision*, *12*(1), 43–77.

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

Briassouli, A., & Narendra, A. (2006). Spatial and Fourier error minimization for motion estimation and segmentation. In *Proceedings of the 18th international conference on pattern recognition, ICPR 2006*, 20–24 August 2006, Hong Kong.

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

Cooke, T. (2008). Two applications of graph cuts to image processing. In *DICTA 2008*.

Dellen, B. K., Clark, J. W., & Wessel, R. (2007). The brain's view of the natural world in motion: Computing structure from function using directional Fourier transformations. *International Journal of Modern Physics B*, *21*(13–14), 2493–2504.

Dellen, B., & Wörgötter, F. (2008). A local algorithm for the computation of optic flow via constructive interference of global Fourier components. In *Proceedings of the British machine vision conference*.

Felzenszwalb, P. F., & Huttenlocher, D. P. (2006). Efficient belief propagation for early vision. *International Journal of Computer Vision*, *70*(1).

Fleet, D. J., & Jepson, A. D. (1990). Computation of component image velocity from local phase information. *International Journal of Computer Vision*, *5*, 77–104.

Galvin, B., McCane, B., Novins, K., & Mills, S. (1998). Recovering motion fields: An evaluation of eight optical flow algorithms. In *Proc. British machine vision conf* (pp. 195–204).

Gautama, T., & Van Hulle, M. (2003). A phase-based approach to the estimation of the optical flow field using spatial filtering. *IEEE Transactions on Neural Networks*, *13*, 1127–1136.

Heeger, D. J. (1987). Model for the extraction of image flow. *Journal of Optical Society of America A*, *4*, 1455–1471.

Horn, B. K. P., & Schunck, B. G. (1981). Determining optic flow. *AI*, *17*, 185–204.

Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proc. DARPA IU workshop* (pp. 121–130).

Mota, C., Stuke, I., & Barth, E. (2001). Analytic solutions for multiple motions. In *Proc. IEEE ICIP* (pp. 917–920). Thessaloniki.

Nagel, H. H. (1987). Displacement vectors derived from second-order intensity variations in image sequences. In *CGIP* (Vol. 21, pp. 85–117).

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

Shizawa, M., & Mase, K. (1990). Simultaneous multiple optic flow estimation. In *Proc. IEEE ICVPR*.

Simoncelli, E. P. (1993). *Distributed representation and analysis of visual motion*. PhD Thesis, Dept. of Electrical Engineering and Computer Science, MIT.

Singh, A. (1990). An estimation-theoretic framework for image-flow computation. In *Proc. IEEE ICCV (Osaka)* (pp. 168–177).

Vernon, D. (1998). Decoupling Fourier components of dynamic image sequences: a theory of signal separation, image segmentation, and optical flow estimation. In *Lecture notes in computer science*. *Computer vision—ECCV'98* (p. 69). Berlin: Springer.

Weickert, J., & Schnörr, C. (2001). A theoretical framework for convex regularizers in PDE-based computation of image motion. *International Journal of Computer Vision*, *14*(3), 245–264.

Weiss, Y., & Freeman, W. T. (2001). On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, *47*(2), 723–735.

Wilson, R., & Granlund, G. H. (1984). The uncertainty principle in signal processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *6*.