Navier-Stokes formulation for modelling turbulent optical flow

Ashish Doshi and Adrian G. Bors

Dept. of Computer Science, University of York, York YO10 5DD, UK
{adoshi,adrian.bors}@cs.york.ac.uk

Abstract

This paper proposes a physics-based methodology for the analysis of optical flows displaying complex patterns. Turbulent motion, such as that exhibited by fluid substances, can be modelled using fluid dynamics principles. Together with supplemental equations, such as the conservation of mass, and well formulated boundary conditions, the Navier-Stokes equations can be used to model complex fluid motion estimated from image sequences. In this paper, we propose to use a robust kernel which adapts to the local data geometry in the diffusion stage of the Navier-Stokes formulation. The proposed kernel is Gaussian and embeds the Hessian of the local data as its covariance matrix. The local Hessian models the variation of the flow in a certain neighbourhood. Moreover, we use a robust statistics mechanism in order to eliminate the outliers from the estimation process. The proposed methodology is applied on artificial vector fields and in image sequences showing atmospheric and solar phenomena.

1 Introduction

Classical optical flow estimation methods work on the assumption that image intensity structures are approximately constant under motion [1, 8]. Robust estimation employing either median statistics or diffusion has been used to eliminate outliers from the optical flow [4] and to smooth colour images while preserving edges [3], respectively. Recently, robust statistics and diffusion have been embedded in a smoothing kernel for jointly processing the data statistics and the local geometry in noisy optical flows [6]. This method was shown to preserve data characteristics as well as the boundaries of the moving objects, while resulting in smoothed optical flows.

Very often, the natural phenomena modelling involve the motion of dynamic fluids which differs radically from that of rigid bodies. Classical optical flow estimation algorithms would fail in such cases. The use of fluid flow modelling for motion estimation can be traced back to the work of Fitzpatrick [7], who compared optical and fluid flow methods. The computation of flows depends largely on the specific nature of the application. Using Fitzpatrick's analysis as a basis, Song and Leahy [12], employed the equation of continuity as an additional constraint to Horn and Schunck's algorithm [8] in order to obtain better motion estimation of the beating heart. Navier-Stokes equations

BMVC 2007 doi:10.5244/C.21.97

have been extensively studied in fluid mechanics for modelling the behaviour of fluids under various conditions and constraints [9]. The Navier-Stokes and optical flow constraint equations have been employed for modelling Karman flows in [10]. Bertalmio *et. al.* applied the Navier-Stokes equations to image and video inpainting [2]. Their approach uses the vorticity-stream formulation of the fluid flow equation, which can be attributed to the image intensity-Laplacian relationship. Corpetti *et. al.* used the vorticity-stream formulation to recover dense motion of water vapours [5].

Navier-Stokes equations have been used in computer graphics for visualising flames and building animation tools based on fluid-like motion [11, 13, 14]. The stable fluid solver (SFS) algorithm implements Navier-Stokes equations and consists of a set of consecutive processing steps [13], such as: advection, diffusion and mass conservation. The boundary conditions are important in constraining the fluid motion [9]. The boundaries have been processed as a set of constraints on a grid [14], by enforcing repetition and employing the Fast Fourier Transform (FFT) [13] or by using level sets [11]. In this study, we extend the SFS solver methodology and apply it for smoothing vector fields estimated from image sequences representing turbulent moving fluids. In our approach, the diffusion step is anisotropic and robust by considering a median of the Hessian diffusion kernel [6]. The proposed hybrid SFS method processes the local geometry and data statistics consistently with the flow motion. The proposed approach is applied for smoothing artificial vector fields and in two image sequences. The paper is structured as follows: Section 2 outlines the SFS algorithm, while Section 3 describes our hybrid solver applied for modelling vector fields. Experimental results and their analysis are presented in Section 4, while Section 5 concludes the paper.

2 The Stable Fluid Method

Navier-Stokes methodology represent the basis for modelling a large variety of phenomena such as those characterising weather, ocean currents, water flow in a pipe, the air flow around a wing, the motion of stars inside a galaxy, blood flow, economics behaviour, etc [9]. In engineering, they are used in the analysis of the effects of pollution, the design of aircraft and of power stations, etc. Navier-Stokes methodology has been applied in Computer Graphics in order to visualise and create the effects given by the complex movement of fluids such as that of coloured gases, air, clouds, liquids, smoke, fire, etc., [11, 13]. The explicit model is generally used for precise computation of fluid dynamics and involves heavy computational complexity [9]. The Von Neumann's stability analysis, as shown in [9], highlights that the implicit model of discretisation when calculating Navier-Stokes equations is unconditionally stable, although it requires a complex numerical implementation scheme. The SFS algorithm proposed by Stam represents an implementation of the Navier-Stokes methodology in an implicit scheme [13, 14].

In order to achieve visual effects, the Navier-Stokes equations are used for both density and velocity in the SFS algorithm [13, 14]. Unlike in the original SFS approach, in this study we consider only the modelling of motion based on the Navier-Stokes equations. The area of investigation (in our case an image or a segmented region from an image) is split into cells located on a grid and we associate a particle to each grid location. Let us assume that the SFS system moves the particles around according to a vector field, where each vector corresponds to a grid location. The Navier-Stokes equation for a given system is derived using the conservation of mass, momentum, and energy for an

arbitrary control volume [9] and is given by :

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{\nabla P}{\rho} + \nu\nabla^2\mathbf{u} + \mathbf{f} \tag{1}$$

where the change of velocity $\mathbf{u}$ over time is represented with respect to the advection, gradient of the pressure P, diffusion and external forcing function $\mathbf{f}$, while ν is a viscosity constant that characterises the fluid and ρ is a parameter. The pressure is assumed to be constant in the given field and its gradient is zero, *i.e.* the change in pressure from one spatial position to another in the vector field is negligible. Consequently, the equation employed by the SFS method is :

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} + \nu\nabla^2\mathbf{u} + \mathbf{f} \tag{2}$$

The diffusion term $\nu\nabla^2\mathbf{u}$ characterises fluids which are assumed incompressible and Newtonian. Moreover, for incompressible fluids it is important to enforce the conservation of mass [9]:

$$\nabla\cdot\mathbf{u} = 0 \tag{3}$$

which states that the divergence of velocity components is zero for infinitesimal time steps. The density of a particle is constant between iterations, thereby the total mass of the field is conserved within the given region.

```
    for k ← 1 to ▷ convergence / number of iterations
        do
1           add force: u_1 = u_0 + f Δt
2           advect: u_2(x) = adv(u_1(x, −Δt))
3           transform: û_2 = FFT(u_2)
4           diffuse: û_3(z) = û_2(z)/(1 + νΔtk^2)
5           conserve: û_4 = conserve(û_3)
6           transform: u_4 = FFT^{-1}(û_4)
```

Figure 1: The stable fluid solver algorithm.

The SFS algorithm proceeds to calculate the velocity components $\mathbf{u}$ as described in Fig. 1, [13]. For each iteration, the first step consists of adding the external forcing function $\mathbf{f}$ which determines the initial conditions in the processing cycle. The second step represents the advection term in equation (2), which corresponds to the following :

$$(\mathbf{u}\cdot\nabla)\mathbf{u} = \left(u_x\frac{\partial u_x}{\partial x} + u_y\frac{\partial u_x}{\partial y}, u_x\frac{\partial u_y}{\partial x} + u_y\frac{\partial u_y}{\partial y}\right) \tag{4}$$

where $\mathbf{u} = (u_x, u_y)$. The analysis of the advection process in real physical phenomena is provided in [9]. The process described by equation (4) is known as the self-advection of velocity. The advection step from the SFS algorithm is implemented by moving the motion vector of each grid cell back in time with $-\Delta t$ by backtracking the velocity field. The third step transforms the velocity field to the frequency domain using the Fast Fourier Transform (FFT). The requirement to set specific boundary conditions is eliminated by extending the spatial repeatability of the area under consideration and by applying FFT. The diffusion term (fourth step) represents the decay of high spatial frequencies in the velocity field and is computed in the Fourier domain with a Gaussian filter processing the velocity component $\mathbf{u}$ by using the time step Δt and the fluid kinematic viscosity ν. The finite difference implicit scheme is used here to discretise the diffusion term in

order to obtain an unconditionally stable system [13]. The fifth step enforces the local incompressibility of the optical flow which requires that the amount of flow entering in a specific area should be equal with the flow exiting that area. The final step projects the flow back from the frequency domain to the spatial-time domain using the inverse FFT transform. This algorithm was modified in [14] by replacing the FFT transformations and the processing in the frequency domain with defining a set of boundary constraints on a grid-based representation of the flow.

3 The Robust Hybrid Fluid Solver

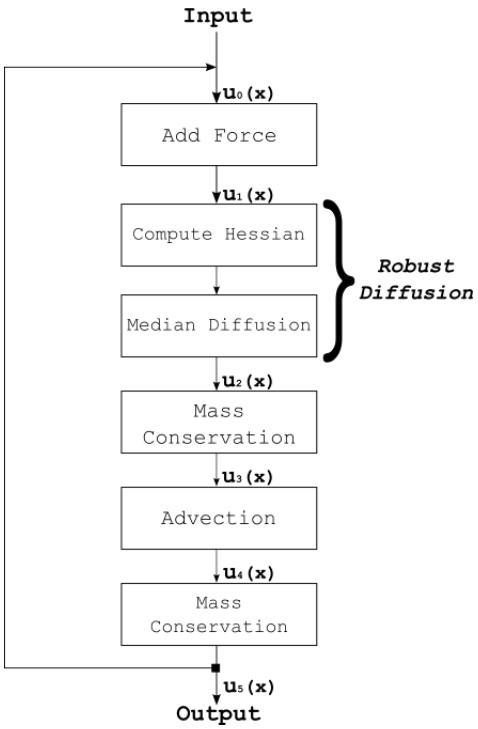


Figure 2: Robust hybrid solver.

The implementation of the stable fluid solver [13] provided rather poor performance in modelling turbulent optical flow estimated from image sequences. This is mainly caused due to the uncertainty in the initial estimation of the optical flow which leads to noise, particularly in image sequences displaying complex motion. In order to improve the performance on optical flow, we propose to embed a robust anisotropic kernel [6] in the diffusion step of the SFS. Fig. 2 shows a flow diagram of the proposed robust hybrid fluid solver. The initial flow can be estimated using the block matching algorithm as in [4] or other motion estimation algorithms [1]. Optical flows provided by block-matching or by using temporal gradient estimation are invariably noisy [4], particularly in the case of image sequences representing moving fluids or other complex phenomena.

The first processing block corresponds to a reinforcement step and in the proposed method is implemented by adding a proportion of the velocity from the previous iteration

to the current velocity :

$$\mathbf{u}_1(t+\Delta t) = (1-\varepsilon)\mathbf{u}_0(t) + \varepsilon\Delta t\mathbf{u}_5(t) \tag{5}$$

where $\mathbf{u}_5(t)$ is the motion vector from the previous iteration t, $\varepsilon \in (0,1)$ is a weighting factor modelling the degree of the reinforcement and $\mathbf{u}_0(t), \mathbf{u}_1(t+\Delta t)$ represent the motion vector reinforced by force at times t and $t+\Delta t$, respectively. At the first iteration there is no reinforcement, *i.e.* $\varepsilon = 0$. The SFS algorithm described in Section 2 proposes to advect the initial flow at Step 2 from Fig. 1. However, that algorithm produces unreliable estimation when applied to noisy vector fields. The optical flow should have a degree of smoothing before advection can be applied. In our approach, we propose to diffuse the noisy flow before proceeding to the advection stage. The transfer function of the original smoothing algorithm is a Gaussian function appropriately defined within the frequency domain [13]. In our approach, we propose to implement a Hessian based diffusion that jointly processes the local geometry and the statistics of the local vector field as in [6] :

$$\hat{\mathbf{u}}_2(t+\Delta t) = \frac{\sum\limits_{\mathbf{x}_i \in \eta(\mathbf{z}_c)} \mathbf{u}_{1,i}(t) \exp[-(\mathbf{x}_i - \mathbf{z}_c)^T \mathbf{H}^{-1} (\mathbf{x}_i - \mathbf{z}_c)]}{\sum\limits_{\mathbf{x}_i \in \eta(\mathbf{z}_c)} \exp[-(\mathbf{x}_i - \mathbf{z}_c)^T \mathbf{H}^{-1} (\mathbf{x}_i - \mathbf{z}_c)]} \tag{6}$$

where $\hat{\mathbf{u}}_2(t+\Delta t)$ is the intermediate diffused value, $\mathbf{H}$ represents the local Hessian, $\mathbf{u}_{1,i}(t)$ is the vector at location i within a neighbourhood $\eta(\mathbf{z}_c)$, centred at the location $\mathbf{z}_c$. The Hessian of the optical flow is calculated locally as :

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 \mathbf{u}}{\partial x^2} & \frac{\partial^2 \mathbf{u}}{\partial x \partial y} \\ \frac{\partial^2 \mathbf{u}}{\partial y \partial x} & \frac{\partial^2 \mathbf{u}}{\partial y^2} \end{bmatrix} \tag{7}$$

The eigenvector corresponding to the largest eigenvalue shows the local direction of the optical flow. This diffusion kernel is anisotropic and adapts to the local structure of the optical flow. Significant optical flow transitions are detected and consequently not smoothed over by the Hessian-based kernel. However, anisotropic diffusion does not deal properly with outliers as shown in a study provided in [6]. In order to properly process the local statistics and eliminate outliers, the median algorithm is considered for robustifying the Hessian based diffusion in the neighbourhood $\eta(\mathbf{z}_c)$.

At the advection stage, our model is only concerned with the nonlinearity of the advection term from equation (4). As mentioned in the previous Section, the self-advection term represents the ability of the velocity components to move their own values from one position to another on a grid in a time step interval, Δt. This procedure involves interpolating the velocity at the grid points, using a neighbourhood approximation, from the previous time step back to the position in the current time step [14].

The model is dependent on the initialisation and on boundary conditions of the system under study. Boundary condition are specifically provided onto the grid in order to represent the physical limits of the optical flow. Such boundary conditions can be the result of image or motion segmentation algorithms or of *a priori* information about the image sequence. There are two boundary conditions to consider. The first condition is determined by the physical boundary. This is represented by the Von Neumann condition which specifies the normal component of the flow to the boundary surface as :

$$\left.\frac{\partial \mathbf{u}}{\partial \mathbf{n}}\right|_{\Omega} = 0 \tag{8}$$

where Ω represents the boundary and $\mathbf{n}$ is its surface normal. This means that the wall absorbs any flow particles coming towards it. For the sake of reducing the required computation complexity, the walls of the domain, Ω are represented by zero values on a geometric grid, which are enforced at every stage of the computation in order to preserve the stability and integrity of the numerical calculation. Since our proposal incorporates both explicit and implicit finite differencing schemes, it is absolutely imperative that the model adheres to the stability criteria, given by $\Delta t/(\Delta \mathbf{x})^2 \leq 1/2$, where $\Delta \mathbf{x}$ represents the location change during the time interval Δt.

The second condition relates to the conservation of mass of the velocity field. The conservation of mass, given by equation (3), should be maintained in order to ensure the incompressibility of the flow. In order to maintain a divergence free velocity field for every stage of computation, the conservation of mass is enforced after both diffusion and advection stages. The conservation of mass stage corresponds to a data normalisation process. The conservation of mass is enforced by using the Helmholtz-Hodge decomposition [13] of the velocity field. This decomposition provides an exact solution so that the mass conserved incompressible flow can be obtained by extracting the gradient of the flow from the current vector field. This decomposition maintains the incompressibility and smoothness of the estimated velocity field. Mass conservation is important for realistically estimating optical flow of fluids. For exemplification, the Helmholtz-Hodge decomposition of the exact closed cavity laminar flow (artificial data experiment provided in Section 4) at the 1000th iteration is shown in Fig. 3.

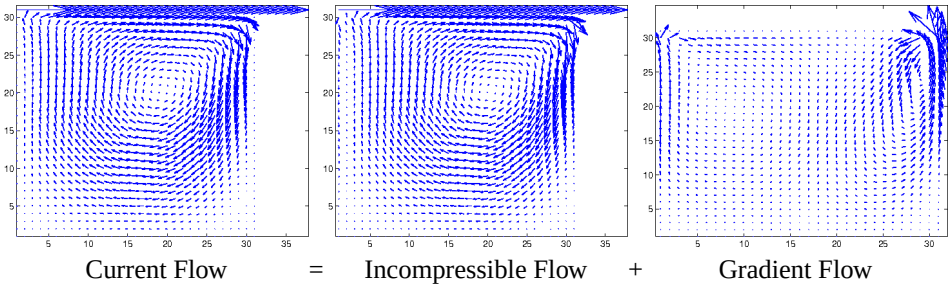
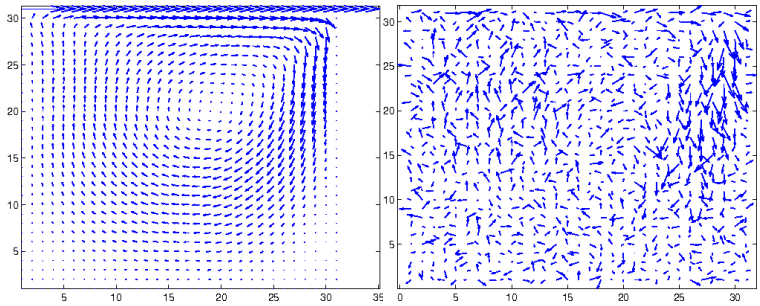


Figure 3: Helmholtz-Hodge decomposition of a closed lid driven cavity laminar flow.

4 Experimental Results

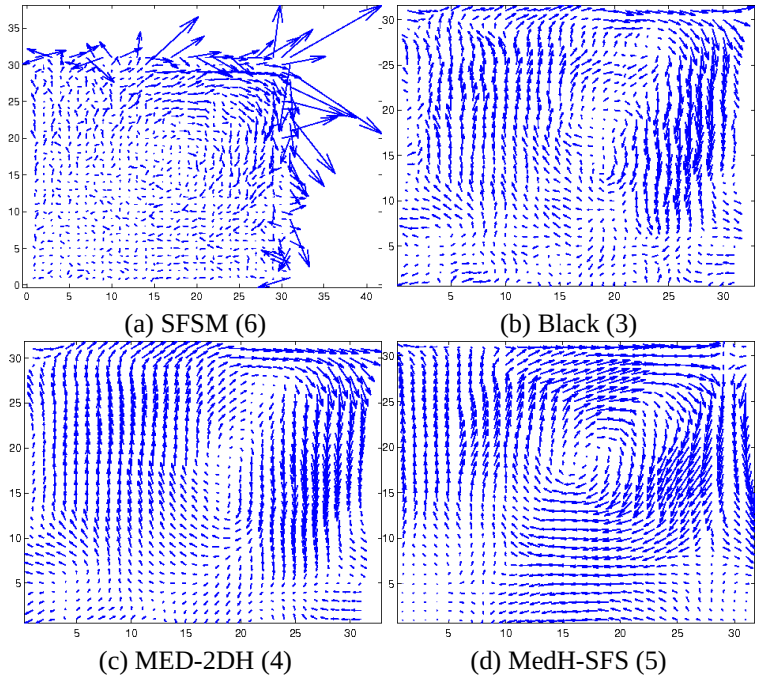
We present results when the proposed algorithm is evaluated on a synthetic vector field and on the optical flow estimated from two real-world image sequences. The synthetic sequence is created using the original Navier-Stokes equations [9] depicting the air flow generated within a lid driven closed cavity. The synthetic flow is created using the vorticity-stream formulation of the Navier-Stokes equations instead of the classic velocity-pressure formulation. Fig. 4(a) represents the simulated synthetic field that visualises the air flow moving with a fixed velocity from left to right inside the top area of a closed cavity. This flow has been obtained after applying the Navier-Stokes equation for a thousand iterations. Fig. 4(b) shows flow degradation after adding Gaussian noise with zero mean and variance $\sigma^2 = 0.25$. Modelling results using the modified SFS (SFSM) algorithm [14] adapted for usage on vector fields is shown in Fig. 5(a), while vector field smoothing using Black's anisotropic diffusion algorithm [3] is provided in Fig. 5(b). Fig. 5(c) shows

the effects of using MED-2DH which is a robust Hessian based diffusion algorithm described in [6], while the robust hybrid fluid solver embedding the median of 2D Hessian diffusion kernel (MedH-SFS) algorithm, as described in Section 3, is shown in Fig. 5(d).



(a) Ground truth synthetic flow (b) $\sigma^2 = 0.25$

Figure 4: Synthetic closed lid-driven cavity flows



(a) SFSM (6) (b) Black (3)

(c) MED-2DH (4) (d) MedH-SFS (5)

Figure 5: Artificial vector field smoothing comparisons. For better visualisation, the vector from the upper-right corner of the SFSM vector field in (a) has been rescaled.

The results in Fig. 5 are obtained at convergence when the mean square error difference between vector fields at two successive iterations is less than 0.01. The number of iterations necessary to achieve convergence is provided in the parentheses from the caption of each result plot of Fig. 5. From these results, we can observe that the vector field modelled by SFSM is still noisy at convergence, while the noise has been significantly reduced in the other smoothed vector fields. It can be observed that MedH-SFS provides the best results and the flow vortex recovered is better located when compared to the vortices recovered using Black and MED-2DH.

Gaussian Noise (σ^2)	SFSM	SFS	MedH-SFS	Black	MED-2DH
0.01	0.7525	0.6211	**0.7634**	0.7226	0.7383
0.10	0.6020	0.5616	**0.7327**	0.6554	0.6997
0.25	0.4538	0.4523	**0.6849**	0.5584	0.6424
0.30	0.4373	0.4624	**0.6704**	0.5567	0.6058
0.40	0.4005	0.4184	**0.5799**	0.4958	0.5556

Table 1: Mean cosine error (MCE) of smoothed vector fields.

For numerical comparisons, we consider the mean cosine error (MCE) between the recovered smoothed flow and the ground truth flow. The MCE is calculated as:

$$\text{MCE} = \frac{\sum_{i=1}^{L} \mathbf{u}_i \cdot \hat{\mathbf{u}}_i}{\|\mathbf{u}_i\| \, \|\hat{\mathbf{u}}_i\| \, L} = \frac{\cos(\theta_i)}{L} \tag{9}$$

where L is the total number of vectors, $\mathbf{u}_i$ is the ground truth before considering the noise and smoothing, and $\hat{\mathbf{u}}_i$ is the result achieved after smoothing the noisy vector field at location i. The MCE is the normalised dot product between two vectors which provides the cosine of the angle between them, denoted as θ_i. The closer MCE is to 1.0, the more similar are the two vector fields. The MCE results are provided in Table 1 after one iteration of smoothing. SFS algorithm was described in Section 2 and was adapted from [13], while SFSM was described in [14]. Both these algorithms have been adapted to work on vector fields. It can be observed that SFS provides good results for a vector field corrupted with low noise variance. However, its performance deteriorates significantly when the noise increases, because the corrupted vector field departs significantly from the Navier-Stokes underlying model. The robust diffusion hybrid fluid algorithm MedH-SFS provides better results than either SFS or SFSM methods in terms of MCE when considering additive Gaussian noise as it can be observed from Table 1. MedH-SFS is also consistently better than Black [3] and MED-2DH [6] anisotropic smoothers.

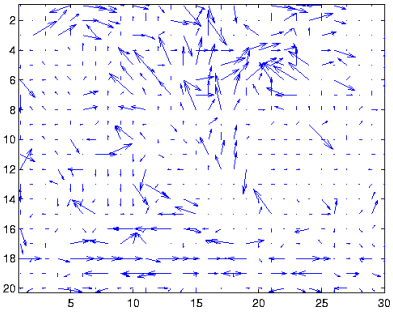
We have applied the proposed methodology on optical flows estimated from image sequences. Fig. 6(a) represents a frame from "Tornado" image sequence, while Fig. 6(b) shows a frame from the "Solar Flare" sequence obtained from Kanzelhöhe Obervatory's solar and environmental research website. The first sequence represents a complex atmospheric phenomenon while the second image is used to observe and analyse solar surface activity. The initial optical flows have been estimated using block matching algorithm (BMA) and are shown in Fig. 6(c) and Fig. 6(d), respectively. The complexity of the motion in the scenes as well as the compression artefacts influence negatively the performance of the BMA algorithm. Fig. 6(e) and Fig. 6(f) show the smoothing result when using MedH-SFS algorithm on the optical flow estimated from the "Tornado" sequence and from the "Solar Flare" optical flow, respectively, both after one iteration. The improvements provided by the Med-SFS over the initial optical flows are significant. We can clearly identify the moving twister and its boundaries after using the proposed methodology as it can be observed in the optical flow from Fig. 6(e). Turbulent movements of the solar surface can be properly identified in Fig. 6(f).

5 Conclusions

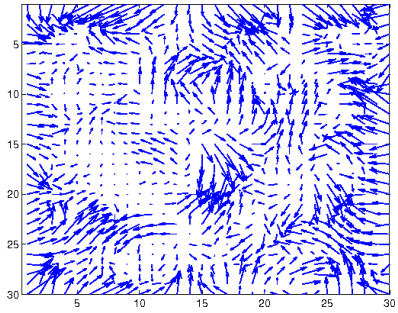
We have presented a physics based model that smoothes and models optical flow representations estimated from images representing complex and turbulent fluid motion. The

(a) Original “Tornado” frame 341

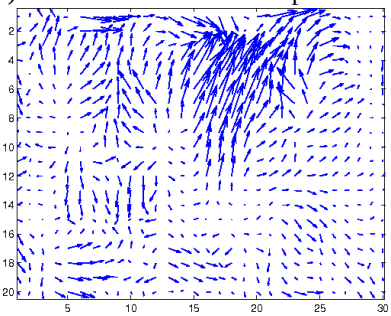
(b) Original “Solar Flare” frame 220



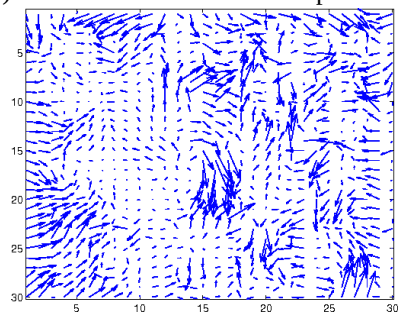
(c) Initial BMA “Tornado” optical flow



(d) Initial BMA “Solar Flare” optical flow



(e) MedH-SFS smoothed “Tornado” flow



(f) MedH-SFS smoothed “Solar Flare” flow

Figure 6: Smoothing optical flows in image sequences displaying turbulent motion.

Stable Fluid Solver (SFS) model is based on the Navier-Stokes equations for incompressible fluid. The SFS algorithm, originally developed in computer graphics for visualising fluid like movement and for building animation tools, has been modified in order to be used on optical flows. The proposed model is highly efficient and stable under certain conditions. The flow incompressibility condition is achieved by imposing the mass conservation through the Helmholtz-Hodge decomposition. We embed a robust Hessian based kernel in the diffusion step of the Navier-Stokes formulation in order to improve the performance of the proposed method for smoothing vector fields. This kernel ensures that smoothing occurs along the structure of the motion field while maintaining the general optical flow structure and the main optical flow features. The proposed kernel ensures robust statistics capability in order to reduce the impact of outliers and thus to enhance the smoothness of the resulting optical flow. The new model is shown to provide good results

in both artificial data and in optical flow from two image sequences, showing turbulent atmospheric and solar activity phenomena.

References

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

[2] M. Bertalmio, A. L. Bertozzi, and G. Sapiro. Navier-Stokes, fluid dynamics, and image and video inpainting. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 355–362, Kauai, HI, 2001.

[3] M. J. Black, G. Sapiro, D. H. Marimont, and D. Heeger. Robust anisotropic diffusion. *IEEE Trans. on Image Processing*, 7(3):421–432, 1998.

[4] A. G. Bors and I. Pitas. Optical flow estimation and moving object segmentation based on median radial basis function network. *IEEE Trans. on Image Processing*, 7(5):693–702, 1998.

[5] T. Corpetti, *É*. Mémin, and P. Pérez. Dense estimation of fluid flows. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(3):365–380, March 2002.

[6] A. Doshi and A. G. Bors. Robust diffusion kernels for optical flow smoothing. In *Proc. IEEE Workshop on Machine Learning for Signal Processing*, pages 415–420, Maynooth, Ireland, 2006.

[7] J. M. Fitzpatrick. A method for calculating velocity in time dependent images based on the continuity equation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 78–81, San Francisco, CA, 1985.

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

[9] J. D. Anderson Jr. *Computational Fluid Dynamics: The Basics with Applications*. McGraw Hill, 1995.

[10] Y. Nakajima, H. Inomata, H. Nogawa, Y. Sato, S. Tamura, K. Okazaki, and S. Torii. Physics-based flow estimation of fluids. *Pattern Recognition*, 36(5):1203–1212, 2003.

[11] S. Premoze, T. Tasdizen, J. Bigler, A. Lefohn, and R. T. Whitaker. Particle-based simulation of fluids. In *Proc. EUROGRAPHICS, Computer Graphics Forum 22(3)*, pages 401–410, 2003.

[12] S. M. Song and R. M. Leahy. Computation of 3D velocity fields from 3-D cine CT images of a human heart. *IEEE Trans. Medical Imaging*, 10(1):295–306, 1991.

[13] J. Stam. A simple fluid solver based on the FFT. *Journal of Graphics Tools*, 6(2):43–52, 2001.

[14] J. Stam. Real-time fluid dynamics for games. In *Proc. Game Developer Conference*, volume 4, pages 76–92, Mar. 2003.