Proc. VIIth Digital Image Computing: Techniques and Applications, Sun C., Talbot H., Ourselin S. and Adriaansen T. (Eds.), 10-12 Dec. 2003, Sydney

Motion Field Estimation for Temporal Textures

David Edwards, Johnny T. Chang, Lin Shi, and Yizhou Yu

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA

Abstract. This paper presents a novel approach for estimating the flow fields of dynamic temporal textures whose motion differs radically from that of rigid bodies. Our approach is based on a local flow probability distribution function at each pixel using the STAR model and the data from a spatio-temporal neighborhood. The peak of this density function can be regarded as the estimated local flow vector. Our complete algorithm exploits a two-stage process. The first stage of the algorithm applies a simple tensor method to estimate the direction of the optical flow at each pixel in the texture. In the second stage, the flow probability function is used to perform a one-dimensional search along the flow direction to obtain the magnitude. Performance analysis and experiments with real video sequences show that our methods can successfully estimate flow fields.

1 Introduction

Our environment is full of motion. Dynamic phenomena, such as drifting clouds, flowing rivers, and curling smoke help to make that natural world a thing of beauty. Videos containing these dynamic phenomena are called temporal textures. Temporal textures, with two spatial dimensions and one temporal dimension, are also known as *dynamic textures*. They consist of multiple two-dimensional frames that change over time to give the appearance of motion. Temporal textures can be hard to analyze because their non-rigid nature makes image registration and geometry reconstruction difficult. Techniques that can successfully analyze and estimate the motion of temporal textures will lead to better algorithms for computer vision and image processing, and in particular, will assist the synthetic generation of temporal textures [1–4] within a scene.

The problem of determining the optical flow, or image-plane velocity, for regular objects is a well-studied problem. Nevertheless, solving optical flow robustly for arbitrary fluids and temporal textures still remains as a challenging problem. In time series analysis, there are some models that have shown potential to model the behavior of temporal textures. The *autoregressive*(AR) model has been often used for predicting the value of a state variable $\boldsymbol{s}$ at time t. The forecast is a linear combination of some number, m, of previous values:

$$\boldsymbol{s}_t = A_1\boldsymbol{s}_{t-1} + \cdots + A_m\boldsymbol{s}_{t-m} + \boldsymbol{n}_t \qquad (1)$$

where the A_t are $r \times r (\boldsymbol{s} \in \boldsymbol{R}^r)$ matrix constants which characterize the sequence. The term $\boldsymbol{n}_t$ is drawn from a zero-mean noise distribution, typically taken to be Gaussian.

This AR model can be generalized to include a spatial neighborhood as well. The result is the spatio-temporal autoregressive model (STAR) which does prediction using a linear combination of the values in a spatio-temporal neighborhood:

$$\boldsymbol{s}(x,y,t) = \sum_{i=1}^{m} A_i \boldsymbol{s}(x+\Delta x_i, y+\Delta y_i, t+\Delta t_i) + \boldsymbol{n}(x,y,t) \tag{2}$$

where Δx_i, Δy_i and Δt_i specify the neighborhood structure of the model. The STAR model only exploits first and second-order statistics, hence cannot model curved lines.

This paper presents new methods for optical flow determination based on these models. The motivation for our methods come from our need for an accurate optical flow determination method in our work on synthesis of temporal textures.

Related Work. Optical flow determination methods can be divided into several classes. *Differential methods* assume the image intensity is a differentiable function and first order approaches such as Newton-Raphson iterations can converge. The most widely used of these differential methods is the Lucas-Kanade algorithm, explained in [5]. *Phase-based methods* use the phase information in the frequency domain. Fleet and Jepson [6] developed an algorithm to compute optical flow based on local phase information in an image. Larsen, Conradsen, and Ersbøll came up with a phase-based method using spatio-temporal Gabor filters, which they apply to meteorological images in [7]. A third class of optical flow determination methods is the class of *tensor methods*. Knutsson explains in [8] how the three-dimensional structure of a scalar field can be represented by a tensor; for temporal textures, eigenanalysis of this tensor yields information about the optical flow.

Several optical-flow determination methods have been developed specifically to work with temporal textures of fluids. Béréziat *et al.* in [9], Corpetti *et al.* in [10, 11] both offer methods for generating optical flow vector fields based on the physical properties of fluid flow mechanics. These physically-based techniques work well for textures where 3D fluid properties are preserved or distortion due to perspective projection is negligible. A scenario where these techniques have been applied successfully is estimating cloud motion from meteorological images for which an orthographic or weak perspective camera model is sufficient. We do not exploit these physical properties in our method because we deal with general cases where pixel intensities are not related to volumetric properties and perspective distortion may be significant. For an excellent survey of optical flow determination methods, see [12].

2 Local Flow Probability Distribution Functions

In this paper, we still assume that the brightness of a point is a constant over a short period of time. However, a statistical approach for determining flow vectors should be adopted instead since the relative positions of neighboring points change faster in a temporal texture than on a rigid body.

Based on the characteristics of temporal textures, we can build a stochastic motion model. The flow vector at a pixel at a certain time can be considered as a random variable with a probability distribution function. This probability distribution function at a pixel (x_s, y_s) at a certain time t can be represented as,

$$\phi_{x_s,y_s,t}(x_d, y_d), (x_d, y_d) \in \boldsymbol{D}_s, \tag{3}$$

where $\boldsymbol{D}_s$ is the set of destination pixels (which is often a neighborhood of pixel (x_s, y_s)) and $\sum_{(x_d,y_d)\in\boldsymbol{D}_s} \phi_{x_s,y_s,t}(x_d, y_d) = 1$. We assume that pixels in a local spatio-temporal region centered at (x_s, y_s, t) share the same probability distribution function for their flow vectors. The flow vector at each pixel in the local spatio-temporal region can be considered as a random sample from the same distribution. Therefore, the statistics of the flow vectors in the local region can be used as an approximation to this distribution function. Once we have the estimation of the probability distribution function, the flow vector at the center pixel (x_s, y_s) is more likely to be consistent with a vector with a high probability.

Recovering probability distribution functions for flow vectors becomes more achievable than recovering accurate pixelwise flow vectors for each frame. The STAR model appears to be very useful for estimating local distributions. Let us first look at Eq. (2) more carefully. The coefficient A_i indicates the degree of correlation between pixel (x, y) at time t and pixel $(x + \Delta x_i, y + \Delta y_i)$ at time $t + \Delta t_i$. When a causal neighborhood is assumed, intuitively, A_i is proportional to the probability of the fluid at pixel $(x + \Delta x_i, y + \Delta y_i)$ actually ending up at pixel (x, y) after a time interval $-\Delta t_i$. However, if we consider pixel (x, y) as the source of flow instead of destination and assume $\Delta t_i > 0$ for all i, the STAR model still has legitimate physical interpretation where A_i is proportional to the probability of the event that the fluid at pixel $(x + \Delta x_i, y + \Delta y_i)$ actually comes from pixel (x, y).

To estimate the probability distribution function at pixel (x_s, y_s), we need to have an estimation of the probability that the fluid at pixel (x_s, y_s) goes to pixel $(x_s + \Delta x_i, y_s + \Delta y_i)$ at the next frame, which means $\Delta t_i = 1$, for all i. In Eq. (2), $\boldsymbol{s}(x, y, t)$ can be either state variables or appearance variables. Since we keep the brightness constancy assumption, $\boldsymbol{s}(x, y, t)$ can actually be replaced by image pixel intensity. Taking the expectation of the both sides of Eq. (2) and setting $\Delta t_i = 1$, we have

$$\overline{I}(x, y, t) = \sum_{i=1}^{m} A_i \overline{I}(x + \Delta x_i, y + \Delta y_i, t + 1) \tag{4}$$

where A_i becomes exactly the probability that the intensity at pixel $(x + \Delta x_i, y + \Delta y_i)$ in the next frame equals the intensity at pixel (x, y) in the current frame. Therefore,

$$\phi_{x_s,y_s,t}(x_s + \Delta x_i, y_s + \Delta y_i) \approx A_i \tag{5}$$

where $(x_s + \Delta x_i, y_s + \Delta y_i) \in \boldsymbol{D}_s$. The expectation in (4) is taken with respect to camera noise as well as the collection of pixels in a spatio-temporal neighborhood of pixel (x, y, t).

Since what we need to estimate is a distribution function with multiple parameters, the amount of data available from a single pixel (x_s, y_s) is obviously not sufficient. As mentioned previously, the same distribution function is used for describing the statistical behavior of all pixels in a spatio-temporal neighborhood $\boldsymbol{N}_s$ of (x_s, y_s, t), which means we can have an equation similar to Eq. (4) for every pixel in $\boldsymbol{N}_s$. This is a linear equation in the set of distribution coefficients and we assume a Gaussian noise source at every pixel. The distribution coefficients can be solved using the following system of normal equations for least-squares when the number of pixels in $\boldsymbol{N}_s$ is larger than the number of pixels in $\boldsymbol{D}_s$.

$$\mathbf{RA} = \mathbf{B} \tag{6}$$

where $\mathbf{R}$ is a symmetric $m \times m$ matrix with $R_{ij} = R_{ji} = \sum_{(x,y,t)\in\mathbf{N}_s} I(x + \Delta x_i, y + \Delta y_i, t+1) I(x + \Delta x_j, y + \Delta y_j, t+1)$, $\mathbf{A}$ is the coefficient vector, and $\mathbf{B}$ is a vector with $B_i = \sum_{(x,y,t)\in\mathbf{N}_s} I(x + \Delta x_i, y + \Delta y_i, t+1) I(x, y, t)$.

Although the STAR model is linear and inappropriate for global curved flow structures, it can still effectively model all local flows very well since even curved flow structures have a first-order local approximation. The least-squares estimation does not guarantee that $\sum_i A_i = 1$ which is a necessary condition for a distribution function. Thus, Eq. (5) should be adapted to

$$\phi_{x_s,y_s,t}(x_s + \Delta x_i, y_s + \Delta y_i) \approx \frac{A_i}{\sum_j A_j} \tag{7}$$

2.1 Analysis

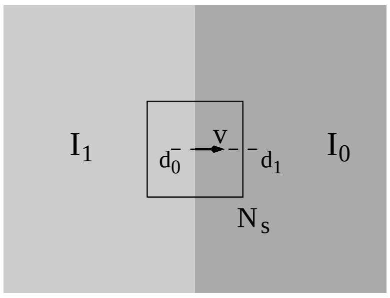

Fig. 1. A step edge moving at a constant velocity v to the right. The image intensity is I_1 to its left, and I_0 to its right. A local marginal flow distribution function is defined on a 1D interval $[d_0, d_1]$, and $\mathbf{N_s}$ is the spatio-temporal neighborhood for least-squares estimation.

Once we have obtained the coefficients in a local flow distribution function, it is natural to consider the true flow vector to be the vector with the maximum value from the distribution function. This corresponds to the maximum likelihood estimation. Let us verify how accurate this estimation is going to be for some simple situations.

Suppose there is a step edge moving at a constant integer velocity, v, across an image from left to right as shown in Fig. 1, and the pixel (x_s, y_s) is right on the edge at time t. A simple $n \times n \times 1$ spatio-temporal neighborhood $\boldsymbol{N}_s$ is centered at (x_s, y_s, t). Because of the aperture problem, we are only able to estimate the normal velocity in the direction perpendicular to the edge. Therefore, only a marginal flow distribution function defined on a one-dimensional window $\boldsymbol{D}_s = [d_0, d_1]$, where $d_0 \leq v \leq d_1$, can be recovered. Without loss of generality, the intensity to the left of the edge is assumed to be I_1, and to the right I_0 with $I_1 > I_0$. Then, in Eq. (6),

$$R_{ij} = R_{ji} = \left(\frac{n}{2} + v - i\right) nI_1^2 + (i-j)nI_0I_1 + \left(\frac{n}{2} - v + j\right) nI_0^2,$$

when $d_1 \geq i \geq j \geq d_0$, and

$$B_i = \begin{cases} \frac{n^2}{2}I_1^2 + (v-i)nI_0I_1 + \left(\frac{n}{2} - v + i\right) nI_0^2, & \text{if } i \leq v; \\ \left(\frac{n}{2} + v - i\right) nI_1^2 + (i-v)I_0I_1 + \frac{n^2}{2}I_0^2, & \text{otherwise.} \end{cases}$$

After matrix simplifications, it can be shown that $\{A_v = 1; A_i = 0, i \neq v\}$ is the unique solution in this case. Therefore, our adapted STAR model can give an accurate optical flow for a moving step edge. Since the solution vector $\mathbf{A}$ resembles a δ function, it is more robust than an estimation from normalized correlation. Note that $\mathbf{R}$ becomes singular when $I_0 = I_1$, which means there is no edge feature in the image any more. We can use this property to detect whether there are sufficient features available.

A more general situation involves a 2D gray-scale pattern moving at a constant velocity, $v = (v_x, v_y)$. Based on the observation that a general 3D spatio-temporal neighborhood centered at $(x_s + v_x, y_s + v_y, t+1)$ is the same as the 3D neighborhood centered at (x_s, y_s, t), we can conclude that the vector $\mathbf{B}$ in Eq. (6) coincides with the j-th column of matrix $\mathbf{R}$ where $\Delta x_j = v_x$ and $\Delta y_j = v_y$. When $\mathbf{R}$ is nonsingular, it is straightforward to verify that $\{A_j = 1; A_i = 0, i \neq j\}$ is still the unique solution, and our method can still return the correct estimation. Obviously, this solution process introduces an internal competition mechanism to let the 'winner' take all. Theoretical analysis for a general situation with nonrigid motion and noise is left for further investigation. In practice, our method has performed well on temporal textures.

2.2 Pruning

Since the motion of temporal textures is largely random and noisy, the flow distribution function needs to be overdetermined during least-squares data fitting. This is not a severe problem since $\boldsymbol{D}_s$ is two-dimensional and $\boldsymbol{N}_s$ is three-dimensional in general. However, pruning can definitely further improve the results. For flow distribution functions, pruning means setting some of their insignificant coefficients to zero before solving the rest of the coefficients. Because of the reduced number of nonzero coefficients, they can be solved more robustly. In Section 3.2, we will consider one-dimensional search as a special type of pruning.

3 A Two-Stage Method

This section explains the details of our two-stage flow determination method. We first explain the *tensor method* for calculating optical flow. Then we discuss how we refine the optical flow estimate from the tensor method to produce a more accurate measurement of the flow vector at each pixel.

3.1 Tensor Method

Instead of working directly with the color values from the texture, we compute the luminance $L(f,x,y)$ for each pixel. The gradient vector of the luminance function at a pixel is $\nabla L = \left(\frac{\partial L}{\partial f}, \frac{\partial L}{\partial x}, \frac{\partial L}{\partial y}\right)$ which points in the direction in which the function changes most rapidly. In practice, we use series-designed first-order derivative filters [12] to estimate the derivative of L along the f-, x-, and y-axes. We can construct a symmetric 3×3 tensor S by taking the outer product of the gradient vector with itself:

$$S = \nabla L \cdot (\nabla L)^T = \begin{bmatrix} \left(\frac{\partial L}{\partial f}\right)^2 & \left(\frac{\partial L}{\partial f}\frac{\partial L}{\partial x}\right) & \left(\frac{\partial L}{\partial f}\frac{\partial L}{\partial y}\right) \\ \left(\frac{\partial L}{\partial f}\frac{\partial L}{\partial x}\right) & \left(\frac{\partial L}{\partial x}\right)^2 & \left(\frac{\partial L}{\partial x}\frac{\partial L}{\partial y}\right) \\ \left(\frac{\partial L}{\partial f}\frac{\partial L}{\partial y}\right) & \left(\frac{\partial L}{\partial x}\frac{\partial L}{\partial y}\right) & \left(\frac{\partial L}{\partial y}\right)^2 \end{bmatrix}$$

This tensor is representative of the local structure of the gray value function. Once we have computed the tensor's value at a pixel, the next step is to calculate the three eigenvalues $|\lambda_{min}| \leq |\lambda_{mid}| \leq |\lambda_{max}|$ and corresponding eigenvectors $\boldsymbol{e}_{min}$, $\boldsymbol{e}_{mid}$, $\boldsymbol{e}_{max}$ of the tensor. This is equivalent to performing a principal component analysis on the texture's gray values. The eigenvector $\boldsymbol{e}_{min}$ corresponding to the eigenvalue λ_{min} of smallest magnitude will point along the direction in which the gray value function changes least quickly. Therefore, this eigenvector $\boldsymbol{e}_{min} = (e_{min_f}, e_{min_x}, e_{min_y})$ is the estimated spatio-temporal velocity at the pixel. To convert this three-dimensional vector $\boldsymbol{e}_{min}$ into a two-dimensional optical flow estimate $\boldsymbol{v}_{tensor}$, we scale the spatial components of the eigenvector by the inverse of the temporal component: $\boldsymbol{v}_{tensor} = \left(\frac{e_{min_x}}{e_{min_f}}, \frac{e_{min_y}}{e_{min_f}}\right)$.

3.2 One-Dimensional Search

Because it is based on spatio-temporal structures, the tensor method is excellent at determining the appropriate direction for the optical flow vector. However, the estimate of the magnitude of the optical flow is inaccurate when the method is applied to temporal textures. After completing the tensor method calculations described above, we have an optical flow estimate $\boldsymbol{v}_{tensor}(f,x,y)$ for a given pixel. In order to refine this estimate, we perform a one-dimensional search.

We apply the method in Section 2 with pruning to this one-dimensional search by constraining the set of nonzero coefficients in the STAR model to the line determined by the tensor method and setting all other coefficients to zero. Therefore, the previous STAR model equation should only include the spatial neighbors, $\{(x + \Delta x_i, y + \Delta y_i)\}$, that lie along the same line. Since we only consider a finite spatial neighborhood in the local flow probability distribution function, in practice, only the set of pixels that lie within the spatial neighborhood as well as on the line can have nonzero coefficients. After solving this small number of coefficients using least-squares, we find the pixel with the highest weight, and estimate the optical flow to be the vector from pixel (f,x,y) to that pixel. Thus, one-dimensional search can effectively reduce the number of unknowns and produce more robust results efficiently.

Having found the best-matching pixels from the previous and next frame, we estimate the optical flow to be the average of the forward and backward optical flows we have obtained from our search.

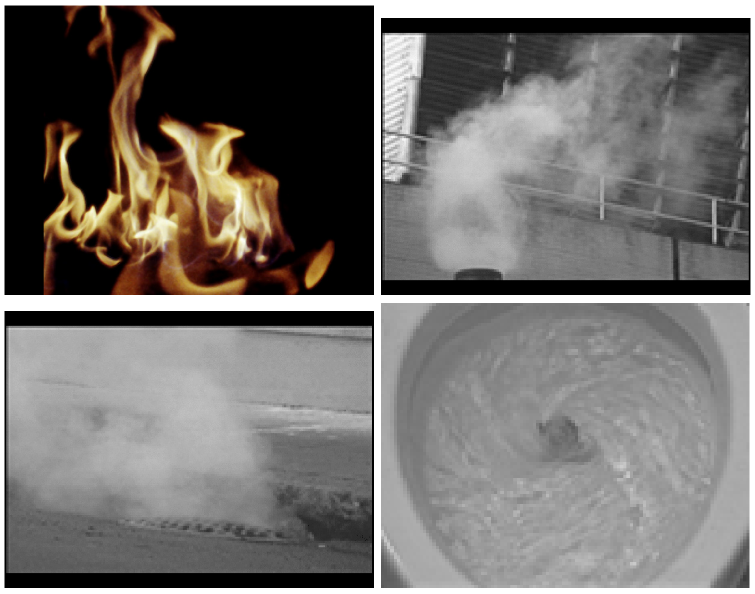

Fig. 2. Example frames from the fire, smoke, steam, and toilet sequences

4 Results

We have conducted experiments on multiple sequences of temporal textures. Most of our input textures came from Martin Szummer's database of temporal texture samples [13]. We used the firergb, smoke, steam, and toilet textures (see figure 2). The firergb texture is a color texture with 100 frames that are 256 pixels wide and 256 pixels high. The other sequences are grayscale textures with frames that are 170 pixels wide and 115 pixels high. The smoke sequence has 150 frames, the steam and toilet textures have 120 frames. All of these results were gathered on a computer with a 650 megahertz Athlon processor and 256 megabytes of RAM.

We ran our optical flow determination code on each of the four sample textures to estimate the flow vector at each pixel. For the sample textures, this took approximately 50 minutes for the steam and toilet sequences, 70 minutes for the smoke sequence, and 2 hours for the firergb sequence. Running time for the algorithm scales linearly with the number of pixels in the texture until the texture becomes large enough that virtual memory becomes an issue.

Figure 3 displays some of the results of our optical flow determination code for the fire, smoke, steam, and toilet sequences. Most of the test image sequences themselves

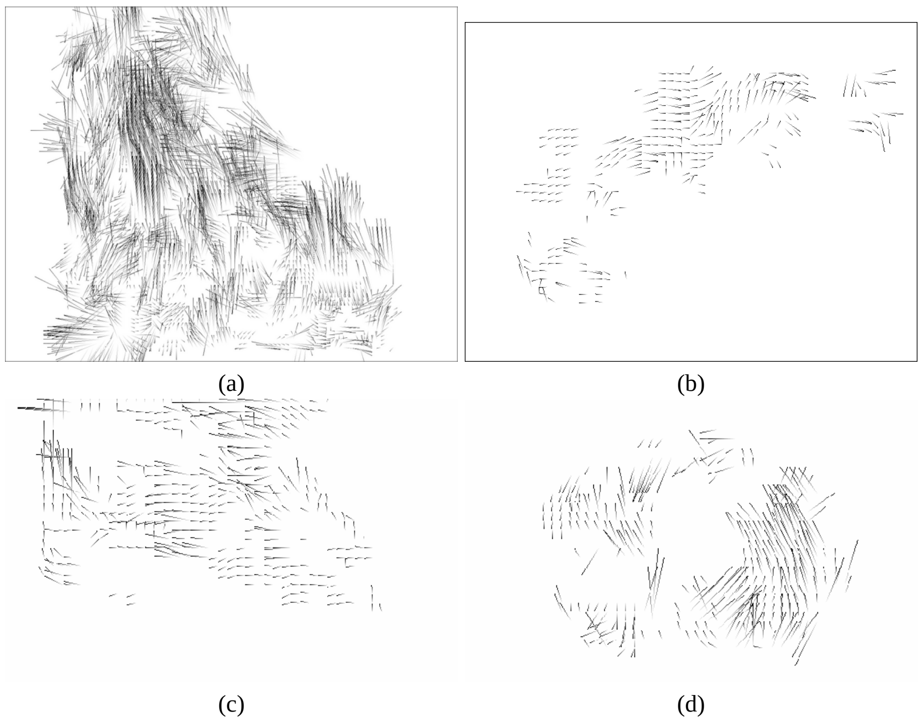

Fig. 3. Optical flow estimates from the (a) fire, (b) smoke, (c) steam, and (d) toilet sequences

provide strong visual cues about the flow directions, therefore, we can visually evaluate the correctness of the results. In general, the optical flow estimates are consistent with human observations.

We have also conducted comparisons between the results from our algorithm and those obtained from three of the previous optical flow algorithms, including Lucas and Kanade's multi-scale algorithm [5], Fleet and Jepson's phase method [6], and an improved version of Quenot *et.al.*'s algorithm [14] whose original version works very well for fluids with artificial particles. We have found that our method performed better than all these algorithms. An example of the comparions is shown in Fig. 4 where a sequence with a falling tidal wave serves as the input. We can see that our algorithm generated much more accurate flow vectors at most of the pixel locations than Lucas and Kanade's and Quenot *et.al.*'s. It also produced more accurate results than the phase method [6] whose results are not visualized here.

5 Conclusions and Future Work

The optical flow determination methods we have presented here produce accurate estimates for many input cases. For gaseous fluids like smoke and fire, the results are very accurate. For liquids such as water, the results are reasonable, but not as good. This may be a result of noise in the image due to the specularity of the water surface.

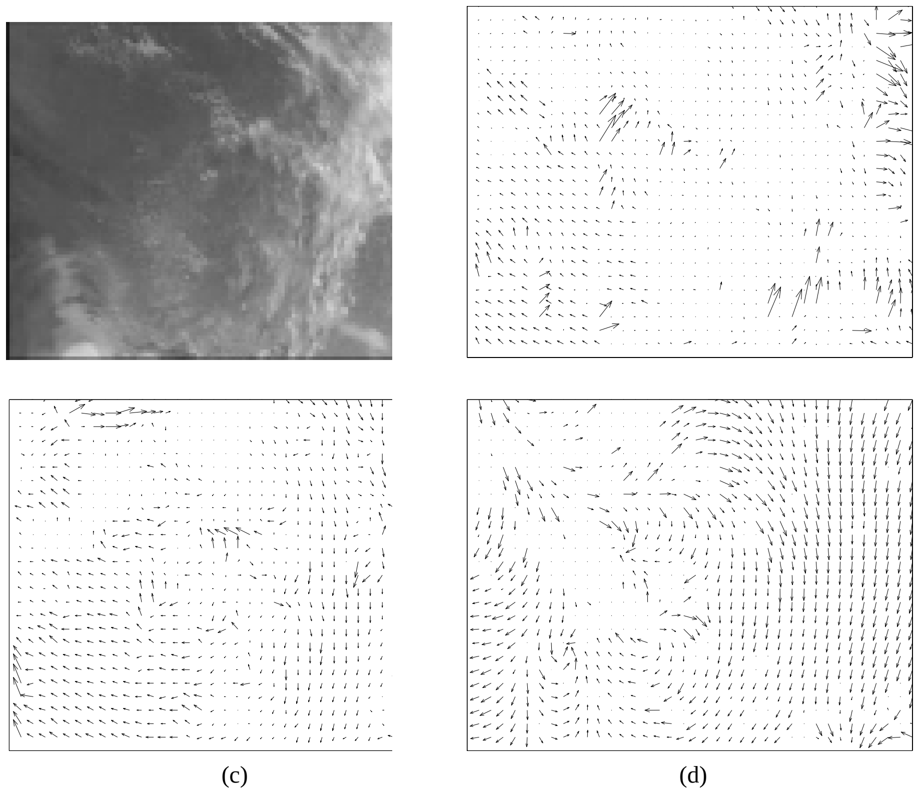

Fig. 4. (a) A tidal wave sequence; (b) flow field from Lucas and Kanade's algorithm; (c) flow field from Quenot *et.al.*'s algorithm; (d) flow field from our algorithm.

Another important fact to take into consideration is that input textures often consist of a fluid superimposed over a background, such as the smoke input texture. It would be interesting to use the optical flow estimates for segmenting the foreground textures from the background.

Our method has a number of uses. We have used it as part of an algorithm for temporal texture synthesis, explained further in [15]. Figure 5 shows frames from a synthesized smoke sequence. For textures with accurate optical flow estimates, our synthesis method was able to produce realistic output textures. By incorporating optical flow information into our synthesis method, we are able to provide not only continuity of color but also continuity of motion in the synthesized texture. This technique allows us to preserve spatial and temporal coherence from an input texture in our synthesized texture.

Acknowledgment

This work was supported by National Science Foundation CCR-0132970.

References

1. Szummer, M., Picard, R.: Temporal texture modeling. In: IEEE International Conference on Image Processing. Volume 3., Lausanne, Switzerland (1996) 823–826
2. Schodl, A., Szeliski, R., Salesin, D., Essa, I.: Video textures. In: Proceedings of Siggraph. (2000) 489–498
3. Bar-Joseph, Z., El-Yaniv, R., Lischinski, D., Werman, M.: Texture mixing and texture movie synthesis using statistical learning. IEEE Transactions on Visualization and Computer Graphics **7** (2001) 120–135
4. Doretto, G., Pundir, P., Wu, Y., Soatto, S.: Dynamic textures. In: IEEE International Conference on Computer Vision. (2001) 439–446
5. Lucas, B., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: The 7th International Joint Conference on Artificial Intelligence. (1981) 674–679
6. Fleet, D., Jepson, A.: Computation of normal velocity from local phase information. In: IEEE Conference on Computer Vision and Pattern Recognition. (1989) 379–386
7. Larsen, R., Conradsen, K., Ersbøll, B.: Estimation of dense image flow fields in fluids. IEEE Transactions on Geoscience and Remote Sensing **36** (1998) 256–264
8. Knutsson, H.: Representing local structure using tensors. In: The 6th Scandinavian Conference on Image Analysis, Oulu, Finland (1989) 244–251
9. Béréziat, D., Herlin, I., Younes, L.: A generalized optical flow constraint and its physical interpretation. In: Conference on Computer Vision and Pattern Recognition, Head Island, South Carolina (2000)
10. Corpetti, T., Mémin, E., Pérez, P.: Dense motion analysis in fluid imagery. In: European Conference on Computer Vision. (2002) 676–691
11. Mémin, E., Pérez, P.: Fluid motion recovery by coupling dense and parametric vector fields. In: International Conference on Computer Vision. (1999) 620–625
12. Jähne, B.: Spatio-Temporal Image Processing: Theory and Scientific Applications. Number 751 in Lecture Notes in Computer Science. Springer (1993)
13. Szummer, M.: Temporal texture sample database (1996) ftp://whitechapel.media.mit.edu/pub/szummer/temporal-texture/raw/.
14. Quenot, G., Pakleza, J., Kowalewski, T.: Particle image velocimetry with optical flow. Experiments in Fluids **25** (1998) 177–189
15. Edwards, D.: Coherence-based temporal texture synthesis using optical flow information. Master's thesis, University of Illinois at Urbana-Champaign (2002)

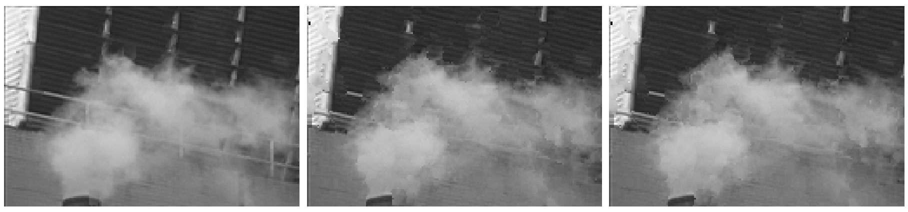

Fig. 5. Frames from a synthesized smoke sequence