Optical Flow at Occlusion

Jieyu Zhang and John L. Barron
Dept. of Computer Science
University of Western Ontario
London, Ontario, Canada, N6A 5B7
barron@csd.uwo.ca

***Abstract*—We implement and quantitatively/qualitatively evaluate two optical flow methods that model occlusion. The Yuan et al. method [1] improves on the Horn and Schunck optical flow method at occlusion boundaries by using a dynamic coefficient (the Lagrange multiplier α) at each pixel that weighs the smoothness constraint relative to the optical flow constraint, by adopting a modified scheme to calculate average velocities and by using a "compensating" iterative algorithm to achieve higher computational efficiency. The Niu et al. method [2] is based on a modified version of the Lucas and Kanade optical flow method, that selects local intensity neighbourhoods, spatially and temporally, based on pixels that are on different sides of an occlusion boundary and then corrects any erroneous flow computed at occlusion boundaries. We present quantitative results for sinusoidal sequence with a known occlusion boundary. We also present qualitative evaluation of the methods on the Hamburg Taxi sequence and and the Trees sequence.**

***Keywords*-2D optical flow, discontinuous optical flow, optical flow at occlusion boundaries, quantitative and qualitative error analysis.**

I. Introduction

The estimation of accurate optical flow in image sequences where occlusion events are occurring is still an open and challenging problem in Computer Vision. Occlusion occurs when an observer see one or more objects moves over other objects or the background in a scene. At such occlusion events there is a motion discontinuity. Standard optical flow methods, such as Lucas and Kanade (LK) [3] or Horn and Schunck (HS) [4] do not model these discontinuities and so compute erroneous flow there. Even more state–of-the-art robust optical flow methods, such as Brox et al. [5], do not produce good flow at occlusion as they do not model occlusion explicitly and may treat the occluding or occluded flow at an occlusion boundary as outliers.

We implement two methods: Yuan et al.'s (Yuan) method [1] is based on the HS method while Niu et al.'s (Niu) method [2] is based on the LK method. These methods were chosen because they explicitly model occlusion, were easy to implement (built on existing MatLab code for LK and HS) and their results were competitive with other recent optical flow occlusion work (see Section II). Because we modify existing MatLab LK and HS code we know that the difference in the Yuan and Niu flow results are entirely due to the additional machinery in the Yuan and Niu methods and not due to any programming artifacts.

We quantitatively evaluate these methods on a synthetic sinusoidal sequence that has an occlusion boundary and where we know ground truth everywhere. We also qualitatively evaluate the flow for two real image sequences with occlusion: the Hamburg Taxi sequence, containing three moving taxis and a walking pedestrian on a stationary background and the Trees sequences (where occlusion occurs at the foreground tree boundaries as Shakey the robot translates). We ran all our experiments on a 2.2GHz machine with 8GB of main memory. The work reported here comprised part of an M.Sc. thesis [6].

II. Literature Survey

Nagel [7] has proposed an "oriented smoothness" constraint that tries to suppress optical flow propagation in the direction orthogonal to the occluding boundaries in an attempt to stop velocity smoothing across motion boundaries. **Bruhn et al.** [8] presented a variational method to compute optical flow in real-time using *bidirectional multigrid strategies*. Their energy functional regularizes a data term and an enhanced HS smoothness constraint that uses piecewise smoothness on both the image intensity distribution and the computed flow. **Deriche et al.** [9] imposes two conditions on the flow to allow discontinuities in the solution: isotropically smoothing the optical flow field inside homogeneous regions and preserving the flow discontinuities in the inhomogeneous regions by smoothing along curves with constant flow but not across them. **Heinrich et al.** [10] used a variational formulation where discontinuities are preserved by non-quadratic regularization using a modified L^p norm. **Schnörr** [11] explicitly represented discontinuities by separating the stationary environment and moving object and then estimating flows over these separate domains. **Heitz** and **Bouthemy** [12] used a multi-modal, coarse-to-fine approach in a global Bayesian decision framework to estimate flow while preserving occlusion discontinuities. **Guichard and Rudin** [13] used the divergence of the flow and modelled occlusion using intensity-based matching and magnitude of divergence constraints. **Ghosal** and **Vaněk** [14] used a 'weighted anisotropic' smoothness term where locations with little available gradient information are more constrained and locations with strong intensity gradients (po-

tential motion discontinuities) are less constrained. **Proesmans et al.** [15] computed flow using non-linear diffusion equations, in which information on optical flow boundaries is non-linearly fed back to the computed flow, preserving the discontinuities. **Thompson** [16] proposed a framework that simultaneously detects flow boundaries and uses those results in flow estimation. Flow analysis for basic Horn and Schunck, Horn and Schunck with boundary detection, edge-based flow estimation with boundary detection and edge-based flow estimation with boundary detection and a projection method were presented. **Güler and Derin** [17] detect discontinuities in noisy and textured images using weak continuity (using an elastic membrane energy model) and line configuration constraints in an graduated non-convexity optimization framework.

In general, it is difficult to quantitatively compare our implementations with previous work because sometimes the images used are unavailable, artificial hand-drawn images were used or no quantitative analysis was performed. Qualitatively, we believe the Yuan et al. and Niu et al. methods produce competitive results with respect to these methods.

III. Experimental Algorithms

We provide a brief introduction to the HS [4] and LK [3] methods, as well as more detailed descriptions of the Yuan et al. [1] and Niu et al. [2] methods.

IV. Horn and Schunck Optical Flow

The HS method minimizes:

$$\int_D (\nabla I^T \cdot \vec{v} + I_t)^2 + \alpha^2(||\nabla u||_2^2 + ||\nabla v||_2^2)dxdy \qquad (1)$$

defined over a domain D (the whole image), where I_x, I_y and I_t are intensity derivatives and α is a Lagrange multiplier that reflects the influence of the smoothness term relative to the optical flow constraint. They used the Gauss-Seidel iterative method to solve the Euler-Lagrange equations for this integral [4] to compute the flow field (u, v). Equation (13) below (with $\omega = 0$) gives those equations.

V. Yuan et al. Optical Flow

Yuan et al. modify the Horn and Schunck method in 3 ways in order to model occlusion. These are described in the next 3 subsections.

A. Modified Smoothing Weight Coefficient

α in Equation (1) weighs the importance between the optical flow constraint and the smoothness constraint in the energy term. The larger α is, the smoother the optical flow field is and the less influence the optical flow constraint has. The smaller α is, the more the flow is consistent the optical flow constraint and the less influence the smoothness term has. Yuan et al. propose that at occlusion boundaries, the smoothness effect should be small while at non-occlusion boundaries and locations the smoothness effect be greater. Thus Yuan et al. propose a modified weight coefficient that varies at each pixel in place of the constant α value HS uses. Yuan et al. compute α_n as:

$$\begin{aligned} \alpha_n &= | \alpha^2 - \alpha^2 \, f(I_x^2 + I_y^2) \times \\ &\quad g((u^n)_x^2 + (u^n)_y^2 + (v^n)_x^2 + (v^n)_y^2) | + \varepsilon, \end{aligned} \qquad (2)$$

where $f(x)$ and $g(x)$ are two threshold functions based on intensity gradients in the images and the velocity gradients in the flow field and n is the image number. $f(x)$ and $g(x)$ are given as:

$$f(x) = \begin{cases} 1 & x \geq \tau_1 \\ 0 & x < \tau_1 \end{cases} \quad \text{and} \quad g(x) = \begin{cases} 1 & x \geq \tau_2 \\ 0 & x < \tau_2. \end{cases} \qquad (3)$$

In Equation (2), the parameter ε is a small positive number used to maintain some smoothness effect along the boundaries of the image. We use $\varepsilon = 0.001$, as suggested by Yuan. The velocity gradients, u_x, u_y, v_x and v_y, are computed at pixel (i, j) using central differences as:

$$u_x = (u_{i+1,j} - u_{i-1,j})/2.0, \qquad (4)$$

$$u_y = (u_{i,j+1} - u_{i,j-1})/2.0, \qquad (5)$$

$$v_x = (v_{i+1,j} - v_{i-1,j})/2.0, \qquad (6)$$

$$v_x = (v_{i,j+1} - v_{i,j-1})/2.0. \qquad (7)$$

With the appropriate selection of τ_1 and τ_2, Yuan et al. claim they can detect pixels on moving boundaries and assign different weight coefficients to these pixels. Yuan et al. suggest values of $\tau_1 = 100.0$ and $\tau_2 = 0.1$.

With the use of varying weight coefficients, Equation (1) changes into two parts. For the pixels on the occlusion boundaries, the energy function becomes:

$$\int_D (\nabla I \cdot \vec{v} + I_t)^2 + \varepsilon(||\nabla u||_2^2 + ||\nabla v||_2^2)dxdy \qquad (8)$$

while for pixels on the background area and inside the moving objects, the energy function becomes:

$$\int_D (\nabla I \cdot \vec{v} + I_t)^2 + (\alpha^2 + \varepsilon)(||\nabla u||_2^2 + ||\nabla v||_2^2)dxdy. \qquad (9)$$

The smoothness constraint at occlusion boundaries is very weak because the small ε inhibits smoothing. At non-occlusion image locations, normal HS velocity diffusion occurs.

In the HS method, the average velocity $\bar{u}$ and $\bar{v}$ in the iterative equations for (u, v) are computed using symmetrical weights, yielding isotropic velocity diffusion. However, in order to avoid spreading the velocities across occlusion boundaries or between an occlusion boundary and the background, Yuan proposes a new way to calculate the average values of velocities. This calculation is motivated by the fact that within a neighbourhood, pixels with smaller intensity

differences should be more important to the average at the central pixel. Averages are now commuted as:

$$\begin{aligned}\bar{u}^n(i,j) = &[e^{-\tilde{f}_{i-1,j}} + e^{-\tilde{f}_{i,j+1}} + e^{-\tilde{f}_{i+1,j}} + e^{-\tilde{f}_{i,j-1}} \\ &\frac{1}{2}(e^{-\tilde{f}_{i-1,j-1}} + e^{-\tilde{f}_{i-1,j+1}} + e^{-\tilde{f}_{i+1,j+1}} + e^{-\tilde{f}_{i+1,j-1}})]^{-1} \\ &\{[e^{-\tilde{f}_{i-1,j}} u^n_{i-1,j} + e^{-\tilde{f}_{i,j+1}} u^n_{i,j+1} + \\ &e^{-\tilde{f}_{i+1,j}} u^n_{i+1,j} + e^{-\tilde{f}_{i,j-1}} u^n_{i,j-1}] + \\ &\frac{1}{2}[e^{-\tilde{f}_{i-1,j-1}} u^n_{i-1,j-1} + e^{-\tilde{f}_{i-1,j+1}} u^n i-1, j+1 + \\ &e^{-\tilde{f}_{i+1,j+1}} u^n_{i+1,j+1} + e^{-\tilde{f}_{i+1,j-1}} u^n_{i+1,j-1}]\}\end{aligned} \tag{10}$$

and

$$\begin{aligned}\bar{v}^n(i,j) = &[e^{-\tilde{f}_{i-1,j}} + e^{-\tilde{f}_{i,j+1}} + e^{-\tilde{f}_{i+1,j}} + e^{-\tilde{f}_{i,j-1}} \\ &\frac{1}{2}(e^{-\tilde{f}_{i-1,j-1}} + e^{-\tilde{f}_{i-1,j+1}} + e^{-\tilde{f}_{i+1,j+1}} + e^{-\tilde{f}_{i+1,j-1}})]^{-1} \\ &\{[e^{-\tilde{f}_{i-1,j}} v^n_{i-1,j} + e^{-\tilde{f}_{i,j+1}} v^n_{i,j+1} + \\ &e^{-\tilde{f}_{i+1,j}} v^n_{i+1,j} + e^{-\tilde{f}_{i,j-1}} v^n_{i,j-1}] + \\ &\frac{1}{2}[e^{-\tilde{f}_{i-1,j-1}} v^n_{i-1,j-1} + e^{-\tilde{f}_{i-1,j+1}} v^n_{i-1,j+1} + \\ &e^{-\tilde{f}_{i+1,j+1}} v^n_{i+1,j+1} + e^{-\tilde{f}_{i+1,j-1}} v^n_{i+1,j-1}]\},\end{aligned} \tag{11}$$

where $f(i,j)$ is the intensity at point (i,j) and

$$\tilde{f}_{i+m,j+n} = \mid f(i+m,j+n) - f(i,j) \mid, \quad (m,n = 1 \; or -1).$$

note that the use of the exponential means large intensity difference at a pixel ensure that pixel has little influence on the neighbourhood average. It is assumed that the intensity gradient at occlusion boundaries tends to sharp, so the new calculation attenuates the effect of spreading velocity from occlusion boundaries to non-boundary parts of the image.

B. Compensating Iterative Algorithm

Yuan et al. propose a compensating iterative method to increase the convergence speed of the iterative equations. This term that takes into account two former iterations (HS only takes into account the last iteration). This relaxation method is described by the following equations:

$$u^{n+1} = \bar{u}^n - \frac{I_x\,[I_x\bar{u} + I_y\bar{v} + I_t]}{(\alpha_n + I_x^2 + I_y^2)} + \omega(u^n - u^{n-1}) \tag{12}$$

and

$$v^{n+1} = \bar{v}^n - \frac{I_y\,[I_x\bar{u} + I_y\bar{v} + I_t]}{(\alpha_n + I_x^2 + I_y^2)} + \omega(v^n - v^{n-1}). \tag{13}$$

Here ω is a relaxation coefficient. Here, we use $\omega = 0.8$ as suggested by Yuan.

VI. Lucas and Kanade Optical Flow

Lucas and Kanade [3] assume constant velocity in local neighbourhoods and solve:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i I_{xi}^2 & \sum_i I_{xi} I_{yi} \\ \sum_i I_{xi} I_{yi} & \sum_i I_{yi}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_{xi} I_{ti} \\ -\sum_i I_{yi} I_{ti} \end{bmatrix} \tag{14}$$

with the sums running from i =1 to n. Sometimes, the derivatives are weighed by values W_i that are inversely proportional to the distance of a pixel from the center pixel. A Gaussian function is typically used to compute such values. From empirical observations [18] it was found that Gaussian weights only improved the solution a negibile amount.

Our implementation of LK uses the smallest eigenvalues of the above integration matrix in Equation (14) to discriminate "good" velocities from "bad" velocities [18]. For each flow calculation, if both eigenvalues, λ_1 and λ_2, are greater than a threshold τ_D, then the velocity is assumed good, otherwise we assume no reliable velocity information can be computed. We use $\tau_D = 1.0$ suggested by Barron et al. [18].

VII. Niu et al. Optical Flow

Niu et al. note that the LK constant flow model is invalid at occlusion boundaries as all pixels from both sides of the occlusion boundary are used in the flow calculation. Niu et al. perform pixel selection in space and time for a neighbourhood, so that only pixels belonging to a single motion are used in a flow calculation. Their method is described in the following four subsections.

A. Spatial Neighbours Selection

Niu et al. do not designate all pixels in neighbourhood with the same importance as LK do. They provide three criteria to select spatial neighbours: intensity differences, spatio-temporal gradient and the large magnitude of temporal derivatives compared to the spatial derivatives.

1) Intensity Differences: The first selection criterion in a spatial neighbourhood is pixel intensity differences. Niu et al. assume that significant intensity differences between adjacent pixels imply object boundaries which should have dis-similar flow. They select pixels within a neighbourhood based on the intensity differences between the central pixel c and other candidate pixels p_i. If the neighbourhood size is $N \times N$ the steps to select the candidates are described as follows:

1) Calculate the intensity difference, d_i, between p_i and c, for $i = 1$ to N^2.
2) Sort the set d_i in ascending order.
3) Select the median value of d_i to be the threshold τ_1 that is used to reject pixels. Hence 50% of pixels are deselected in each neighbourhood.

This adaptive threshold used to select candidates is more accurate than using a constant threshold.

2) Spatio-Temporal Gradient: An inconsistent gradient is also a criterion used to select candidates from the previous step. The spatio-temporal gradient, computed from I_x, I_y and I_t, is the direction of maximum intensity variation and pixels within the neighbourhood should have similar intensity gradient vectors because a smooth variation within a small surface patch is assumed. Thus, those pixels with inconsistent gradient vectors usually imply regions containing object boundaries or other discontinuities. Niu et al. reject pixel p in the window if:

$$\begin{cases} sign\left(\frac{I_x(p)}{I_t(p)}\right) = -\,sign\left(\frac{I_x(c)}{I_t(c)}\right) \\ sign\left(\frac{I_y(p)}{I_t(p)}\right) = -\,sign\left(\frac{I_y(c)}{I_t(c)}\right) \end{cases} \tag{15}$$

or

$$\begin{cases} sign\left(\frac{I_{xt}(p)}{I_{tt}(p)}\right) = -sign\left(\frac{I_{xt}(c)}{I_{tt}(c)}\right) \\ sign\left(\frac{I_{yt}(p)}{I_{tt}(p)}\right) = -sign\left(\frac{I_{yt}(c)}{I_{tt}(c)}\right) \end{cases}, \tag{16}$$

where the $sign$ function is given as:

$$sign(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0. \end{cases} \tag{17}$$

After this step, all the candidates pixels in the window are rejected if their gradient vector are significantly different from the central pixel's. As we can see in Equations (15) and (16) this difference is computed as a sign change of the ratio of 1^{st} and 2^{nd} order intensity derivatives.

3) Occlusion and Non-occlusion Outliers: Usually, if a pixel is at an occlusion event, the optical flow constraint equation is invalid. Thus, the system of equations given by Equation (14), as used in the LK method, is invalid. Niu et al. reject pixels as outliers based on the observation that the temporal derivatives of those pixels tend to have large magnitudes compared to their spatial derivatives. They reject a candidate pixel, p, if it meets the following criteria:

$$\frac{\mid I_t(p) \mid}{\mid I_x(p) \mid + \mid I_y(p) \mid} > T_1, \tag{18}$$

$$\frac{\mid I_{xt}(p) \mid}{\mid I_{xx}(p) \mid} > T_2 \quad \text{and} \tag{19}$$

$$\frac{\mid I_{yt}(p) \mid}{\mid I_{yy}(p) \mid} > T_3, \tag{20}$$

where the threshold T_1, T_2 and T_3 are given by the image averages of the above three ratios.

B. Temporal Neighbours Selection

Niu et al. extend the spatial smoothness constraint to be a spatio-temporal one. They propose the selection of coherent neighbours in the temporal domain based on the flow vector $\mathbf{v}$ obtained in the spatial domain from the LK system of equations $A\mathbf{v} = \mathbf{b}$. They compute a residual error using I_x, I_y and I_t from adjacent images as:

$$r = \mid I_x u + I_y v + I_t \mid .$$

r is expected to have small values if the flow is consistent. In this case, they assume it is safe to use these pixels in their temporal neighbourhood selection. They extend the spatial constraint to include temporal neighbours if r is smaller than a threshold τ_2. Since they do not specify a value for τ_2, we use $\tau_2 = r_0 + 5.0$, where r_0 is the residual error of the current frame's pixel.

The newly selected neighbours form a new system of equations:

$$R\mathbf{v} = \mathbf{d}, \tag{21}$$

where elements of R and $\mathbf{d}$ have the same form as for A and $\mathbf{b}$. The combination of the two systems of equations can be written as:

$$W\mathbf{v} = \begin{bmatrix} A \\ R \end{bmatrix} \mathbf{v} = \begin{bmatrix} \mathbf{b} \\ \mathbf{d} \end{bmatrix}. \tag{22}$$

Again, we use least squares to obtain a solution for $\mathbf{v}$. In our implementation of spatio-temporal selection, we only use one frame before and after the image. Niu et al. did not indicate the number of frames they used.

C. Correction of Erroneous Flow

The selection of spatio-temporal neighbours is based on the validity of the optical flow constraint. When there is a discontinuity, this assumption is no longer valid. If the invalidity is caused by occlusion/disocclusion, they correct for erroneous flow, as described below.

In Equation (22), a discontinuous region can be detected from computed acceleration components, u_t and v_t, which tend to have large magnitudes. Thus, $\mid u_t \mid + \mid v_t \mid > \tau_3$ is the criterion indicating a velocity correction is necessary. We use $\tau_3 = ||(d_u, d_v)||_2$ to compute this value, where d_u and d_v is the velocity difference between the velocities obtained from Equations (21) and (22). The procedure for velocity correction is described by the following steps:

1) Divide the neighbourhood into four overlapping regions: the left half Ω_1, the right half Ω_2, the upper half Ω_3 and the lower half Ω_4;
2) Calculate the average velocity in these four rectangular sub-regions:

$$v_1 = mean(v(p)), p \in \Omega_1, \tag{23}$$

$$v_2 = mean(v(p)), p \in \Omega_2, \tag{24}$$

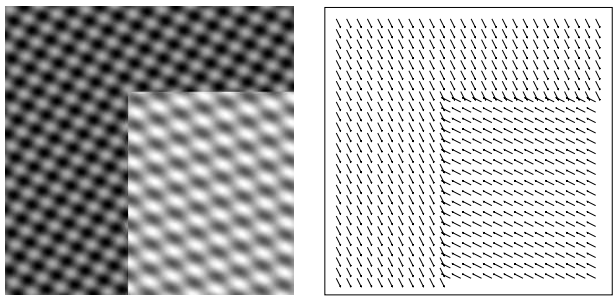
$$u_1 = mean(u(p)), p \in \Omega_3, \tag{25}$$

$$u_2 = mean(u(p)), p \in \Omega_4. \tag{26}$$

3) We form 4 vectors: $\vec{a} = (0, u_1)$, $\vec{b} = (0, u_2)$, $\vec{c} = (0, v_1)$ and $\vec{d} = (0, v_2)$. Let θ_1 be the angle between $\vec{a}$ and $\vec{b}$ and θ_2 be the angle between $\vec{c}$ and $\vec{d}$, If both θ_1 and θ_2 are both positive and the flow magnitude large (> 5) an occlusion event is assumed. If θ_1 and θ_2 are both negative and the flow magnitude small (≤ 5) a non-occlusion event is assumed.
4) For an occlusion region, the preceding frame's derivatives are used in Equation (22) while for a non-occlusion region, the followings frame's derivatives are used in Equation (22).

VIII. Experimental Technique

We used the formula from Barron et al. [18] to generate left and right sinusoidal patterns of the sinusoidal image sequence. The left pattern has constant velocity of $(2, 1)$ while the right pattern has a constant velocity of $(-1, -2)$. The sinusoidal sequence with an occlusion boundary is constructed by combining the left and right patterns so that the left pattern is initially the whole image and the right pattern is pasted over the left pattern so as to have a the vertical boundary at the line $x = 85$ and a horizontal boundary at the line $y = 60$ (with a corner point is at (85,60)). Figures 1a and 1b shows the 9^{th} sinusoidal image and its correct flow respectively.



(a) (b)

Figure 1. (a) The 9^{th} image of the sinusoidal sequence comprised a left sinusoid (the occluding surface) moving with velocity $(2, 1)$ and a right sinusoid (the occluded surface) moving with velocity $(-1, -2)$ and (b) the correct flow field for this image.

We used the Fleet angular error [18] to measure the accuracy of computed flows. If $\vec{v}_c = (u_c, v_c, 1)$ is the correct velocity at the pixel (i, j) and $\vec{v}_e = (u_e, v_e, 1)$ is the estimated velocity at pixel (i, j) we compute the angle between normalized versions of them as an error measure:

$$AE = \arccos(\hat{v}_c \cdot \hat{v}_e). \tag{27}$$

We report average angular error (AAE) plus standard deviation (STD) below.

We used Simoncelli's matched/balanced filters [19] to compute the spatio-temporal image intensity derivatives I_x, I_y and I_t.

IX. Sinusoidal Results

We present quantitative results for the sinusoidal sequence below.

A. Sinusoidal Error Comparison

Quantitative error for the left and right sinusoidal sequences alone for all four methods are quite low, on the order of 0.1° of error (or less) and not reported here. Table I shows the comparison between the HS, Yuan, LK and Niu methods for the sinusoidal sequence. From the results, we can conclude that the Yuan and Niu methods improve the HS and LK methods for the sinusoidal sequence.

Technique	AAE	STD	Density
HS	4.75°	16.75°	100%
Yuan	4.51°	16.78°	100%
LK	8.36°	27.72°	99.23%
Niu	3.81°	16.73°	99.25%

Table I
THE ANGULAR ERRORS AND STANDARD DEVIATIONS FOR FLOW FIELDS WITH 100% DENSITY COMPUTED BY HORN AND SCHUNCK ($\alpha = 10.0$, 100 ITERATIONS), YUAN ET AL. ($\alpha = 10.0$, $\varepsilon = 0.001$, $\omega = 0.8$, 100 ITERATIONS), WITH 99.23% DENSITY COMPUTED BY LUCAS AND KANADE ($\tau_D = 1.0$) AND WITH 99.25% DENSITY COMPUTED BY NIU ET AL.($\tau_D = 1.0$) FOR THE 9^{th} FRAME OF 2DSIN.

Figure 2 shows the computed flow fields for the four methods, respectively. Due to space limitations, we do not show the corresponding error images, but two things are evident from these images: all the error occurs only at the occlusion boundary and the error for LK is by far the worst while the error for Niu is the best.

B. Sinusoid Parameter Variation

We tested the variation of parameter α for Horn and Schunck method (100 iterations) and the variation of parameters α and ω for the Yuan method ($\varepsilon = 0.001$, 100 iterations) on the sinusoidal sequence. We present these 100% dense results in Tables II and III below. From these results we can see that $\alpha = 10.0$ and $\omega = 0.8$ generate the most accurate flow fields.

Technique	AAE	STD
HS ($\alpha = 1.0$)	4.58°	17.30°
HS ($\alpha = 10.0$)	4.75°	16.75°
HS ($\alpha = 20.0$)	5.69°	16.77°
Yuan ($\alpha = 1.0$)	4.54°	17.30°
Yuan ($\alpha = 10.0$)	4.51°	16.78°
Yuan ($\alpha = 20.0$)	4.79°	16.94°

Table II
THE ANGULAR ERRORS AND STANDARD DEVIATIONS FOR FLOW FIELDS WITH 100% DENSITY AND VARIATION OF α COMPUTED BY HORN AND SCHUNCK (100 ITERATIONS), YUAN ET AL. ($\varepsilon = 0.001$, $\omega = 0.8$, 100 ITERATIONS) FOR THE 9^{th} FRAME OF 2DSIN.

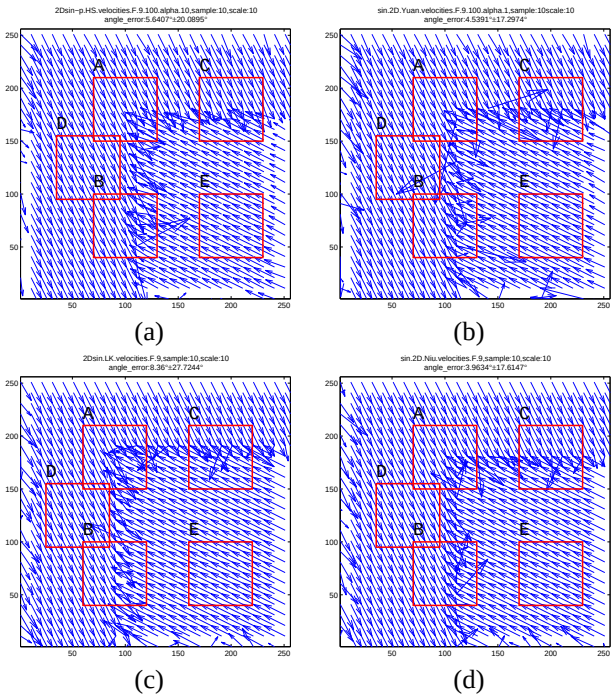


Figure 2. The flow fields computed by (a) HS ($\alpha = 10.0$, 100 iterations) with 100% density, (b) Yuan ($\alpha = 10.0$, $\varepsilon = 0.001$, $\omega = 0.8$, 100 iterations) with 100% density, (c) LK ($\tau_D = 1.0$) with 99.23% density and (d) Niu ($\tau_D = 1.0$) with 99.25% density methods for the 9^{th} frame of the sinusoidal sequence. Note the flow fields have regions A, B, C, D and E delimited.

Technique	AAE	STD
Yuan ($\omega = 0.2$)	4.60°	16.78°
Yuan ($\omega = 0.4$)	4.53°	16.78°
Yuan ($\omega = 0.6$)	4.51°	16.94°
Yuan ($\omega = 0.7$)	4.51°	16.78°
Yuan ($\omega = 0.8$)	4.51°	16.78°

Table III
THE ANGULAR ERRORS AND STANDARD DEVIATIONS FOR FLOW FIELDS WITH 100% DENSITY AND VARIATION OF ω COMPUTED BY YUAN ET AL. ($\alpha = 10.0$,$\varepsilon = 0.001$, 100 ITERATIONS) METHOD FOR THE 9^{th} FRAME OF 2DSIN.

C. Sinusoid Execution Times

Table IV presents the execution time for the four methods running the on sinusoidal sequence. We can see that the spatio-temporal selection of neighbourhoods in Niu's method is time-consuming. We also note that the local flow LK method is by far the fastest (compared to the Niu method or the regularization methods).

D. Sinusoidal Flow Region Analysis

Figure 2 shows the flow fields for the HS, Yuan, LK and Niu methods with five 61×61 regions delimited: A (vertical occlusion boundary), B (corner point of occlusion boundary), C (vertical occlusion boundary), D (left unoccluded pattern) and E (right unoccluded pattern). Since all

Technique	Density	Time
HS ($\alpha = 10.0$)	100%	8.62s
Yuan ($\omega = 0.2$)	100%	15.53s
Yuan ($\omega = 0.8$)	100%	13.77s
LK	99.23%	0.29s
Niu	99.25%	74.96s

Table IV
TIME ANALYSIS FOR SINUSOIDAL SEQUENCE. $\alpha = 10$ AND 100 ITERATIONS WERE USED FOR HS AND YUAN. $tau_D = 1.0$ WAS USED FOR LK AND NIU.

Region Error Analysis			
Method	Region A	Region B	Region C
HS	15.65 ± 35.90	8.97 ± 28.03	8.85 ± 26.68
Yuan	11.88 ± 29.55	8.40 ± 25.94	9.00 ± 25.72
LK	28.81 ± 46.80	20.91 ± 41.99	18.34 ± 40.42
Niu	13.07 ± 33.16	8.67 ± 27.29	7.19 ± 23.49
Method	Region D	Region E	Overall
HS	0.77 ± 0.44	0.06 ± 0.03	5.64 ± 20.09
Yuan	0.05 ± 0.04	0.09 ± 0.05	4.54 ± 17.30
LK	0.06 ± 0.06	0.10 ± 0.05	8.36 ± 27.72
Niu	0.06 ± 0.06	0.10 ± 0.05	3.96 ± 17.62

Table V
AAE ERRORS FOR THE 5 REGIONS, A, B, C, D AND E, AS SHOWN IN FIGURE 2, (PLUS THE OVERALL ERRORS).

error occurs at the occlusion boundary, the overall error is lower than the errors for regions A, B and C. The average angular errors and standard deviations for these regions is given in Table V. We can see the error for regions A, B and C are much higher that for the non-occluded images areas (D and E), where the error is quite low. The corner point error is highest, probably because both I_x and I_y are wrong there, while the horizontal and vertical errors are about the same (One of I_x or I_y only are wrong there). The standard deviations for regions A, B and C are quite high, indicating a lot of intensity derivative variability at occlusion. We see that the error results for Yuan et al. are better than Horn and Schunck while the results for Niu et al. are better than Lucas and Kanade. The Niu et al. results are the best overall while the Lucas and Kanade results are the worst overall.

X. QUALITATIVE ANALYSIS

We use Hamburg Taxi sequence (made by Nagel's research group at the Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe) and Trees sequence (made by Shakey the Robot at Carnegie Mellon University) for qualitative error analysis of the four methods. Figure 3 shows the 9^{th} image of the Hamburg and Trees sequences (no ground truth is available).

A. Hamburg and Trees Error Comparison

Figures 4a, 4b, 4c and 4d show the computed flow fields for Hamburg Taxi scene for HS, Yuan, LK and Niu while Figures 4e, 4f, 4g and 4h show the computed flow fields for the Trees sequence for HS, Yuan, LK and Niu. From the

(a) (b)

Figure 3. (a) The 9^{th} frame of the Hamburg Taxi scene (233×256) and Trees sequence (190×256).

computed flow fields we can see the Yuan method generate more accurate optical flow than the HS and LK methods. The Niu method is marginally better than the LK method but both methods produce poor flow for these sequences. For the regularization methods (HS and Yuan) flow "bleeds" across boundaries, as it does for Nagel's "oriented smoothness" flow [7], [18].

B. Hamburg and Trees Parameter Variation

We tested the variation of parameter α for the HS and Yuan methods for the Hamburg and Tree sequences. Due to space limitations we can not show these flow fields but $\alpha = 10$ generates the qualitatively best looking flow fields for both sequences.

C. Hamburg and Trees Execution Time

We recorded typical execution times for the four methods running on the Hamburg Taxi and Trees sequences. For the 2 sequences, HS required 6.57 and 7.05 seconds respectively, Yuan required 11.80 and 11.73 seconds respectively, LK required 0.22 and 0.31 seconds respectively and Niu required 56.60 and 69.68 seconds respectively. LK has the smallest execution times while Niu had by far the highest execution time. The spatio-temporal neighbourhood selection used by Niu is quite time consuming.

XI. Conclusion

We examined the performance of the Yuan et al. [1] and Niu et al. [2] optical flow methods that explicitly model occlusion. We performed both quantitative and qualitative analysis on these methods and compared the results with those from the HS and LK methods for the sinusoidal sequence (with an occlusion boundary) and for the Hamburg Taxi sequence and the Trees sequences.

Our quantitative results showed that when the HS and Yuan methods are compared, the Yuan method is about 10% to 20% better. The Niu method showed slightly better results for the sinusoidal sequence than the LK method. The LK method is the worse (but the fastest) of the 4 methods. The Yuan method required almost twice the time of the HS method. The compensating algorithm was unable to alleviate the cost of the Yuan machinery added to the basic HS method. Our qualitative analysis show that Yuan and Niu are better that the HS and LK methods and it is worthwhile to explicitly model occlusion.

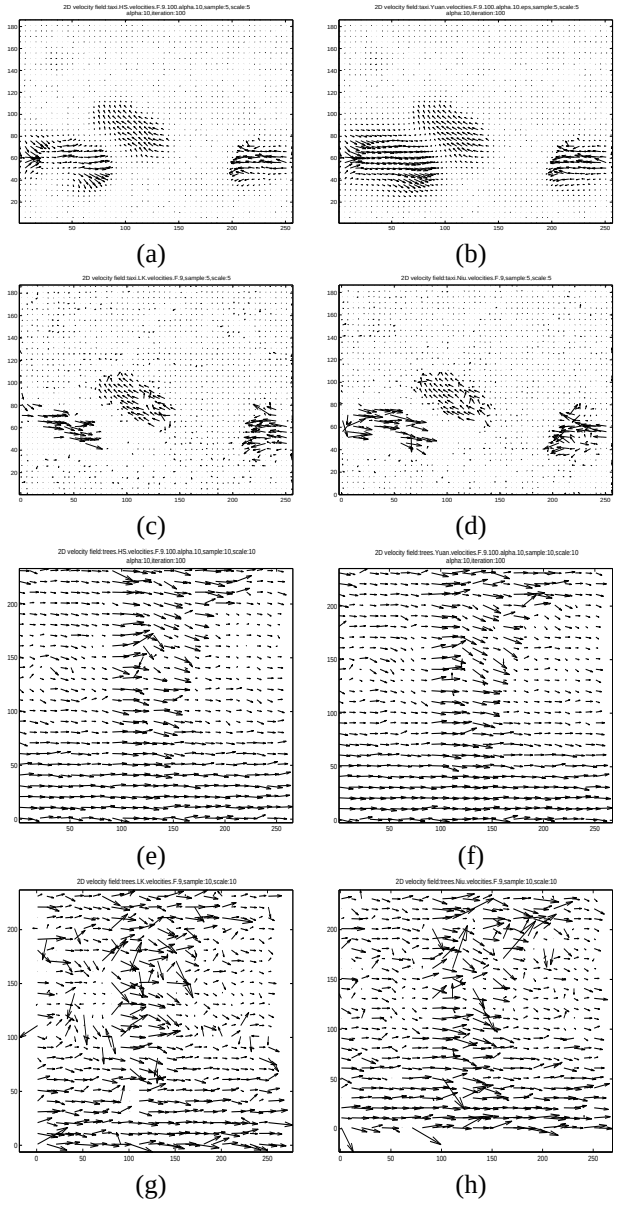


Figure 4. The flow fields computed by (a) and (e) Horn and Schunck ($\alpha = 10.0$, 100 iterations) with 100% density, (b) and(f) Yuan et al. ($\alpha = 10.0$, $\varepsilon = 0.001$, $\omega = 0.8$, 100 iterations) with 100% density, (c) and (g) Lucas and Kanade ($\tau_D = 1.0$) with 72.17% and 97.24% density respectively and (d) and (h) Niu et al. ($\tau_D = 1.0$) with 97.97% and 97.24% density respectively for the 9^{th} frame of Hamburg Taxi and Trees sequences.

A. Future Work

One idea for future work is to detect closed occlusion boundaries first, as described by Stein and Herbert [20] and

Sundberg et al. [21] and then compute robust regularized flow within each closed occlusion boundary segment. Both Stein and Herbert and Sundberg et al. used optical flow in their occlusion boundary detection schemes but they did not show any optical flow results in their papers.

Our current idea would be to use the morphological watershed method (provided by the MatLab IP toolbox) to compute the initial closed boundary segmentation (Stein and Herbert used a non-MatLab watershed method) and then filter out non-occlusion boundaries by detecting which edges are moving (as Sundberg et al. did). When 2 adjacent regions have a common non-occlusion edge eliminated they would be merged. Finally, an optical flow calculation could be performed on each of the remaining closed segments, independently of each other. The optical flow at occluding boundaries would then allow edges and adjacent surfaces to be labelled occluding or occluded.

Another important issue not addressed yet (as far as we know) is what effect a robust calculation would have on occlusion boundary optical flow: are erroneous flow vectors there filtered out by a robust calculation? We expect not, as the robust estimator would be presented with 2 (or more) distributions of different flow vectors belonging to the surfaces sharing the common occlusion boundary. We have noticed that there are some obvious outliers in the region flows in Figure 2 for the sinusoidal flow that we would expect to be eliminated by a robust calculation. One obvious way to investigate robustness would be to recast the 4 methods in a robust framework and analyze the results.

ACKNOWLEDGMENTS

A NSERC Discovery grant supported Jieyu Zhang during her MSc graduate studies.

REFERENCES

[1] L. Yuan, J. Li, B. Zhu, and Y. Qiao, "A discontinuity-preserving optical flow algorithm," in *1st IEEE Intl. Sym. on Systems and Control in Aerospace and Astronautics*, 2006, pp. 450–455.

[2] Y. Niu, A. Dick, and M. Brooks, "Discontinuity-preserving optical flow computation by a dynamic overdetermined system," in *Digital Image Computing: Techniques and Applications*. IEEE Computer Society, 2007, pp. 352–359.

[3] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Intl. Joint Conf. on Art. Intel.*, vol. 3, 1981, pp. 674–679.

[4] B. Horn and B. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[5] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," *ECCV*, pp. 25–36, 2004.

[6] J. Zhang, "Algorithms for 2d and 3d optical flow at occlusion: Quantitative and qualitative results," Master's thesis, Dept. of Computer Science, University of Western Ontario, Dec. 2011.

[7] H.-H. Nagel, "Displacement vectors derived from second-order intensity variations in image sequences," *Computer Vision, Graphics, and Image Processing*, vol. 21, no. 1, pp. 85–117, 1983.

[8] A. Bruhn, J. Weickert, T. Kohlberger, and C. Schnörr, "Discontinuity-preserving computation of variational optic flow in real-time," *Scale Space and PDE Methods in Computer Vision*, pp. 279–290, 2005.

[9] R. Deriche, P. Kornprobst, and G. Aubert, "Optical-flow estimation while preserving its discontinuities: a variational approach," *Recent Developments in Computer Vision*, pp. 69–80, 1996.

[10] M. Heinrich, M. Jenkinson, M. Brady, and J. Schnabel, "Discontinuity preserving regularisation for variational optical-flow registration using the modified lp norm," *Medical Image Analysis for the Clinic: A Grand Challenge*, pp. 185–194, 2010.

[11] C. Schnörr, "Computation of discontinuous optical flow by domain decomposition and shape optimization," *Intl. Journal of Computer Vision*, vol. 8, no. 2, pp. 153–165, 1992.

[12] F. Heitz and P. Bouthemy, "Multimodal estimation of discontinuous optical flow using markov random fields," *IEEE Trans. PAMI*, vol. 15, no. 12, pp. 1217–1232, 1993.

[13] F. Guichard and L. Rudin, "Accurate estimation of discontinuous optical flow by minimizing divergence related functionals," in *IEEE Intl. Conf. on Imaging Processing*, vol. 1, 1996, pp. 497–500.

[14] S. Ghosal and P. Vanek, "A fast scalable algorithm for discontinuous optical flow estimation," *IEEE Trans, PAMI*, vol. 18, no. 2, pp. 181–194, 1996.

[15] M. Proesmans, L. Van Gool, E. Pauwels, and A. Oosterlinck, "Determination of optical flow and its discontinuities using non-linear diffusion," in *ECCV*, 1994, pp. 294–304.

[16] W. Thompson, "Exploiting discontinuities in optical flow," *Intl. Journal of Computer Vision*, vol. 30, no. 3, pp. 163–173, 1998.

[17] S. Güler and H. Derin, "Detection of discontinuities in noisy and textured images using weak continuity constraints," in *IEEE Proc. 36^{th} Midwest Symposium on Circuits and Systems*, 1993, pp. 245–248.

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

[19] E. Simoncelli, "Design of multi-dimensional derivative filters," *IEEE Intl. Conf. Image Processing*, vol. 1, pp. 790–793, 1994.

[20] A. Stein and M. Herbert, "Occlusion boundaries from motion: Low-level detection and mid-level reasoning," *Intl. Journal of Computer Vision*, vol. 82, no. 3, pp. 325–357, 2009.

[21] P. Sundberg, T. Brox, M. Maire, P. Arbeláez, and J. Malik, "Occlusion boundary detection and figure/ground assignment from optical flow," in *CVPR*, Colorado Springs, June 2011, pp. 2233–2240.