

Real-time Performance of Variational Optical Flow Computation

ZHILEI CHAI, ZHIBIN WANG, QIN WU, and JIUZHEN LIANG, Engineering Research Center of Internet of Things Technology Applications Ministry of Education, Jiangnan University

Techniques of optical flow computation are widely used in many video/image based applications such as motion estimation, video compression and so on. Among different techniques, variational methods have the reputation of producing dense optical flow fields. Hence, enhanced variants of variational optical flow methods are being proposed continuously. However, the real time performance of the variational method is becoming a key problem to deal with because of its increasing computational complexity and real-time requirements with the development of embedded vision systems.

In this paper, we investigate key factors affecting the real-time performance of variational methods by theoretical and empirical analysis of several classic methods. The parallelizability of each functional module is also analysed. It is expected that the result of this paper can provide a systematic guidance for implementing the variational optical flow method efficiently, especially in embedded and real-time systems.

Categories and Subject Descriptors: C.2.2 [**Real-Time and Embedded Technology and Applications**]: Real-time computing, operating systems and real-time communications; algorithms for resource management and quality of service control

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: variational optical flow methods, real-time performance, embedded systems, theoretical and empirical analysis

1. INTRODUCTION

Optical flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and the scene. The optical flow computation is used to do the estimation of motion by extracting either instantaneous image velocities or discrete image displacements. Performance evaluation such as [Barron et al. 1994; Baker et al. 2011] shows that variational methods starting from Horn and Schunck [Horn and Schunck 1981] can produce the best performing and understood techniques for optical flow computation. When comparing with the sparse flow estimation method such as Lucas-Kanade [Lucas and Kanade 1981], variational methods are well suited for computing dense optical flow fields [Black and Anandan 1996; Alvarez et al. 2000]. Moreover, they can preserve motion boundaries [Nagel and Enkelmann 1986; Beauchemin and Barron 1995], perform favourably in the presence of noise and occlusion [Black and Anandan 1996; Bruhn et al. 2005; Sundberg et al. 2011]. Recent improvement of variational methods can even treat large displacements correctly [Alvarez et al. 2000; Brox and Malik 2011].

However, variational methods require minimization of the energy function, which is often achieved by solving the linear or non-linear discretization of Euler-Lagrange equa-

This research was partly funded by NSF of China with grant No. 60703106, 61170121, 61202312 and supported by the 111 Project (B12018).

Author's addresses: #1800 of Lihu Ave, Wuxi, Jiangsu Prov, P.R.China; email: zlchai@jiangnan.edu.cn.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© YYYY ACM 1539-9087/YYYY/01-ARTA \$15.00

DOI 10.1145/0000000.0000000 <http://doi.acm.org/10.1145/0000000.0000000>

tion. In order to solve the corresponding equation numerically, classic iterative methods such as Gauss Seidel (GS) and Successive Over Relaxation (SOR) are frequently used [Young 1971]. Sometimes thousands of iterations would be necessary to achieve convergence. In addition, many pixel-wise processing like smoothing or gradient computation are also needed for extracting basic information. All computation above are super time consuming. That is why variational optical flow methods are difficult to be processed in real-time.

Presently, there have been a few attempts to improve the real-time performance of variational methods to meet the requirement especially in embedded systems. One is accelerating the convergence of iteration by employing methods such as multi-grid schemes [Bruhn and Weickert 2006; Bruhn 2006]. The other is accelerating optical flow methods by computing them in real parallel through specific hardware platforms. For instance, when implemented on GPU Nvidia GTX 480 [Sundaram et al. 2010], the variational method of Brox [Brox and Malik 2011] can be processed within 1.1 seconds for one frame of image with resolution 640×480 . When combining hardware acceleration with the multi-grid scheme, the real-time performance can be improved further. For instance, [Gwosdek et al. 2010] implements the linear combined-local-global (CLG) method [Bruhn et al. 2005] with the multi-grid scheme on Sony PlayStation 3. It can be processed at the frame rate of 110 fps for the video sequence with resolution 316×252 .

With the development of embedded vision systems and related applications, the real-time performance of variational optical flow computation becomes critical because it will be used more often in embedded systems interacting with the real world. Furthermore, the image resolution is increasing continuously which will need more processing time. Currently, the frame rate and resolution can be processed in real-time by variational optical flow methods are still far from practical. However, what are the key factors for the real-time performance and which methods are suitable choice under some situation still have not been analysed clearly. Thus, a systematic study for real-time performance of variational optical flow methods should be considered.

The goal of this paper is to find out key factors affecting the real-time performance of variational methods. It is carried out by analysing several classic variational methods both from theoretical and empirical aspects. The parallelizability of each functional module is also analysed. It is expected that the result of this paper can provide a systematic guidance for implementing the variational optical flow method efficiently, especially in embedded and real-time systems.

The rest of the paper is organized as follows. Section 2 discusses the related work of variational optical flow methods from the aspects of result accuracy and real-time performance respectively. Section 3 introduces three kinds of classic variational optical flow algorithms including Horn and Schunck Model [Horn and Schunck 1981], Combine Local and Global Model [Bruhn et al. 2005] and Combine Brightness and Gradient Model [Brox et al. 2004]. Improved techniques such as non-quadratic penalization [Weickert and Schnorr 2001], coarse-to-fine warping technique [Black and Anandan 1996; Brox et al. 2004; Papenberg et al. 2006] are also introduced. Section 4 analyses the computational properties of basic functional modules extracted from algorithms above. Section 5 demonstrates experimental results and gives the empirical analysis about computational time corresponding to the methods and modules in section 4. Section 6 draws conclusions and tries to predict the future work to do. Additionally, the corresponding experimental results about flow fields of different methods are also presented in the appendix for the purpose of comparison.

2. RELATED WORK

Variational optical flow methods determine the desired displacement field by the minimizer of a suitable energy function, where deviations from model assumptions are

penalized. In general, the energy function consists of two terms (i.e., data term and smoothness term) along with some constraints.

Since the classic work of Horn and Schunck [Horn and Schunck 1981] was proposed in 1981, lots of enhanced variants of variational optic flow methods have been introduced. All of these variants can be viewed as some kinds of improvement of the energy function or its terms. Generally, the goal of these variants is to obtain better flow fields or decrease the computational complexity. Thus, we are going to introduce these variants as follows:

(1) **Variants for Improving the Flow Field Result.**

- **For Data and Smoothness Term.** Allowing for piecewise smoothed results and discontinuities in the flow field, the quadratic regulariser in Horn and Schunck model is replaced by the smoothness constraint [Nagel and Enkelmann 1986; Black and Anandan 1996; Weickert and Schnorr 2001]. In order to deal with outliers in the model assumptions, the quadratic penalize is replaced by non-quadratic robust functions in the smoothness assumption [Brox 2005; Bruhn 2006]. The same outlier problem is also dealt with in data term [Black and Anandan 1996]. In order to deal with non-constant illumination, the gradient constancy assumption [Brox et al. 2004] is elaborately investigated as a cue that is invariant against brightness changes. However, the gradient constancy assumption can only provide information at positions of the image where second order derivatives are not zero. Thus a new constraint called combining brightness and gradient assumptions is proposed [Brox et al. 2004].
- **For the Energy Function.** Generally, local methods are more robust under noise, while global techniques yield dense flow fields. So it attempts to combine advantages of Lucas-Kanade [Lucas and Kanade 1981] and Horn and Schunck [Horn and Schunck 1981], so called the combining local and global optical flow method [Bruhn et al. 2005]. The idea to refrain from a linearization of the data constancy assumption is not completely new. Brox proposed a multi-resolution strategy called the non-linearised constancy assumption [Brox et al. 2004], which not only yields improved results, but also shows strong similarities to so-called warping methods [Brox et al. 2004; Papenberg et al. 2006; Zimmer et al. 2011].

(2) **Variants for Improving the Real Time Performance.**

- **Improving Algorithm.** For variational methods, it has to solve a large and sparse linear or non-linear Euler-Lagrange equation. A broad class of efficient solvers have been derived [Young 1971]. Gauss Seidel (GS) and Successive Over Relaxation (SOR) represent the class of non-hierarchical iterative solvers. Unidirectional coarse-to-fine techniques represent the class of simple hierarchical iterative schemes [Black and Anandan 1996; Memin and Perez 1998]. Full multi-grid (FMG) methods are known to deliver the best performance among all multi-grid solvers [Bruhn 2006; Briggs et al. 2000]. Gwosdek [Gwosdek et al. 2010] suggests a multi-grid red-black relaxation used in a parallel implementation of the linear 2D-CLG [Bruhn et al. 2005] method.
- **Hardware Acceleration.** A GPU implementation of non-linear 2D-CBG [Brox et al. 2004] was proposed using the Jacobi solver [Grossauer and Thoman 2008] which achieves a frame rate up to 17 fps with the resolution of 511×511 . Another GPU implementation [Sundaram et al. 2010] can compute high quality LDof [Brox and Malik 2011] with the resolution of 640×480 within 1.8 seconds.

3. VARIATIONAL OPTICAL FLOW COMPUTATION METHODS

In this section, the theoretical framework of variational optical flow methods is described. For clarity, a general energy function is introduced first. If $D^k \mathbf{I}$ denotes the set of partial (spatial or temporal) derivatives with order k , then the energy function can be formulated as

$$E(\mathbf{u}) = \int_{\Omega} (M(D^k \mathbf{I}) + \alpha S(\nabla \mathbf{I}, \nabla \mathbf{u})) d\mathbf{x} \quad (1)$$

where $M(D^k \mathbf{I})$ denotes the data term that represents one or more constancy assumptions on $D^k \mathbf{I}$ given in their original non-linear or their linearised form. $S(\nabla \mathbf{I}, \nabla \mathbf{u})$ denotes the smoothness term assuming that the optic flow field \mathbf{u} is smooth or piecewise smooth. For preserving continuities, this term can be replaced by the image gradient $\nabla \mathbf{I}$ or the flow gradient $\nabla \mathbf{u}$.

For better understanding, we take three classic methods as instances to introduce their data term, smoothness term and energy function in more detail. The selected methods includes the Horn and Schunck Model [Horn and Schunck 1981], the Combine Local and Global Model [Bruhn et al. 2005; Bruhn and Weickert 2006], and the Combine Brightness and Gradient Model [Brox et al. 2004]. Among all methods, the Combine Local and Global Model is the pioneer to improve the real time performance of variational optical flow.

3.1. Protocol A: Horn and Schunck Model (HS)

This model uses the brightness constancy assumption as the data term and the homogeneous regularization as the smoothness term. In order to formulate assumptions mathematically, the optic flow vector $w(\mathbf{x}) := (u(\mathbf{x}), v(\mathbf{x}), h_t)$ between two successive frames of an image sequence $I : (\Omega \subset \mathbb{R}^3) \rightarrow \mathbb{R}$ is used. $\mathbf{x} = (x, y, t)$ is a point in the spatio-temporal domain with t denoting the time axis. The grid size h_t in temporal direction is set to 1. Assuming that the grey value stays constant from the pixel (x, y) in the frame at time t to the shifted pixel $(x + u, y + v)$ in a successive frame at time $t + 1$, it yields the constraint

$$I(x, y, t) = I(x + u, y + v, t + 1). \quad (2)$$

In order to solve u and v , this non-linear equation is usually linearised by a first order Taylor expansion. It leads to a linearised grey value constancy assumption. That is so-called the optic flow constraint (OFC)

$$I_x u + I_y v + I_t = 0 \quad (3)$$

where subscripts denote partial derivatives. Evidently, the equation itself is not sufficient to determine two unknowns u and v uniquely. It is overcome by assuming that the flow field is globally smooth. This assumption can be formulated mathematically by penalizing large spatial flow gradients ∇u and ∇v . Thus, the Horn and Schunck Model combines the brightness constancy assumption and the smoothness assumption in a single variational framework.

3.1.1. A1: Quadratic Data and Smoothness Terms. The energy function of the original Horn and Schunck [Horn and Schunck 1981] is described as follows

$$E(u, v) = \int_{\Omega} \underbrace{((I_x u + I_y v + I_t)^2)}_{\text{Data}} + \alpha \underbrace{(|\nabla u|^2 + |\nabla v|^2)}_{\text{Smoothness}} dxdy. \quad (4)$$

To minimize the energy function, we can obtain the following Euler-Lagrange equations

$$(I_x u + I_y v + I_t) I_x - \alpha(\Delta u) = 0$$

$$(I_x u + I_y v + I_t) I_y - \alpha(\Delta v) = 0 \quad (5)$$

where Δ denotes the operator of Laplace 2D-filter. The discrete Euler-Lagrange equations at each pixel (i, j) of the image can finally be written as

$$\begin{aligned} [I_x^2]_{i,j} [u]_{i,j} + [I_x I_y]_{i,j} [v]_{i,j} + [I_x I_t]_{i,j} - \alpha \left(\sum_{l \in (x,y)} \sum_{(\tilde{i}, \tilde{j}) \in N_l(i,j)} \frac{[u]_{\tilde{i}, \tilde{j}} - [u]_{i,j}}{h_l^2} \right) &= 0 \\ [I_x I_y]_{i,j} [u]_{i,j} + [I_y^2]_{i,j} [v]_{i,j} + [I_y I_t]_{i,j} - \alpha \left(\sum_{l \in (x,y)} \sum_{(\tilde{i}, \tilde{j}) \in N_l(i,j)} \frac{[v]_{\tilde{i}, \tilde{j}} - [v]_{i,j}}{h_l^2} \right) &= 0 \end{aligned} \quad (6)$$

for $i = 1, \dots, N$ and $j = 1, \dots, M$ on a rectangular grid with spacing h_l in l -direction, where $N_l(i, j)$ denotes the set of neighbours of pixel (i, j) in the direction of l -axis. Equations above are linear equations of $[u]_{i,j}$ and $[v]_{i,j}$, which can be solved quite efficiently by the SOR method [Young 1971].

3.1.2. A2: Non-quadratic Data and Smoothness Terms. The quadratic penalty gives too much influence to outliers. To allow discontinuities in the optical flow field (e.g., object boundaries), they should be penalized less severely. Hence, the usage of non-quadratic penalty functions in data and smoothness assumptions is proposed [Black and Anandan 1996; Weickert and Schnorr 2001]. The L1 norm [Brox et al. 2004; Bruhn et al. 2005; Wedel et al. 2008; Zimmer et al. 2011] is a common choice of the penalty function like $\Psi(s^2) = \sqrt{s^2 + \varepsilon^2}$ with $\varepsilon = 0.001$ that is corresponding to Total Variational (TV) regularization. Both data and smoothness term are applied with the robust function. The energy function is written as

$$E(u, v) = \int_{\Omega} \underbrace{(\Psi((I_x u + I_y v + I_t)^2))}_{Data} + \underbrace{\alpha \Psi(|\nabla u|^2 + |\nabla v|^2)}_{Smoothness} dx dy. \quad (7)$$

In contrast to the quadratic penalty, this model contains the assumption of a piecewise smooth flow field and accepts outlier discontinuities in the optical flow field. With a non-quadratic penalize function in both data and smoothness terms, Euler-Lagrange equations are no longer linear on u and v . Consequently, it needs to remove the non-linearity by means of a fixed point iteration scheme in the Euler-Lagrange equation. There are two iterations: the inner loop solves the linear system and the outer one removes or updates the non-linearity. Hereby, we also use the expression shown in Equation 6 to denote four neighbours of the pixel (i, j) . It may contain less than four elements when considering the boundary of an image. In the corresponding linear discretion of Euler-Lagrange, equations in the $k + 1$ th iteration, with k denoting the fixed point iteration, is given by

$$\begin{aligned} &[\Psi'_D]_{i,j}^k [I_x^2]_{i,j} [u]_{i,j}^{k+1} + [\Psi'_D]_{i,j}^k [I_x I_y]_{i,j} [v]_{i,j}^{k+1} + [\Psi'_D]_{i,j}^k [I_x I_t]_{i,j} \\ &- \alpha \sum_{l \in (x,y)} \sum_{(\tilde{i}, \tilde{j}) \in N_l(i,j)} \frac{([\Psi'_S]_{\tilde{i}, \tilde{j}}^k + [\Psi'_S]_{i,j}^k) ([u]_{\tilde{i}, \tilde{j}}^{k+1} - [u]_{i,j}^{k+1})}{2 h_l^2} = 0 \\ &[\Psi'_D]_{i,j}^k [I_x I_y]_{i,j} [u]_{i,j}^{k+1} + [\Psi'_D]_{i,j}^k [I_y^2]_{i,j} [v]_{i,j}^{k+1} + [\Psi'_D]_{i,j}^k [I_y I_t]_{i,j} \\ &- \alpha \sum_{l \in (x,y)} \sum_{(\tilde{i}, \tilde{j}) \in N_l(i,j)} \frac{([\Psi'_S]_{\tilde{i}, \tilde{j}}^k + [\Psi'_S]_{i,j}^k) ([v]_{\tilde{i}, \tilde{j}}^{k+1} - [v]_{i,j}^{k+1})}{2 h_l^2} = 0 \end{aligned} \quad (8)$$

where non-linear abbreviations $[\Psi'_D]_{i,j}$ and $[\Psi'_S]_{i,j}$ denote the functions $\Psi'_D(s^2)$ and $\Psi'_S(s^2)$ evaluated at the pixel (i, j) , i.e.

$$\begin{aligned} [\Psi'_D]_{i,j} &= \frac{1}{2\sqrt{([I_x]_{i,j}[u]_{i,j} + [I_y]_{i,j}[v]_{i,j} + [I_t]_{i,j})^2 + \varepsilon_D^2}} \\ [\Psi'_S]_{i,j} &= \frac{1}{2\sqrt{[\|\nabla u\|]_{i,j}^2 + [\|\nabla v\|]_{i,j}^2 + \varepsilon_S^2}}. \end{aligned} \quad (9)$$

$[\|\nabla u\|]_{i,j}^2$ denotes the gradient magnitude operator suggested by Bruhn et al [Bruhn 2006; Bruhn and Weickert 2006], i.e.

$$[\|\nabla u\|]_{i,j}^2 = \frac{1}{2} \left[\left(\frac{[u]_{i+1,j} - [u]_{i,j}}{h_x} \right)^2 + \left(\frac{[u]_{i-1,j} - [u]_{i,j}}{h_x} \right)^2 + \left(\frac{[u]_{i,j+1} - [u]_{i,j}}{h_y} \right)^2 + \left(\frac{[u]_{i,j-1} - [u]_{i,j}}{h_y} \right)^2 \right]. \quad (10)$$

For non-quadratic terms, it should be noted that the non-linearity of the preceding system is now hidden in expressions $[\Psi'_D]_{i,j}$ and $[\Psi'_S]_{i,j}$ which actually depend on $[u]_{i,j}$ and $[v]_{i,j}$. Similarly, linearised equations above can also be solved by the SOR method [Young 1971].

3.1.3. A3: Non-linear Non-quadratic Data and Smoothness Terms. As introduced in [Brox et al. 2004; Papenberg et al. 2006], the idea of the non-linearisation of data constancy assumption is not new. It is coupled with a multi-resolution strategy that yields improved results but worse real time performance in contrast to two methods above.

Applying non-quadratic penalize functions to both the data and the smoothness term is also integrated with the non-linear brightness constancy assumption. It is straightforward to derive an energy function

$$E(u, v) = \int_{\Omega} \underbrace{(\Psi(|I(\mathbf{x} + \mathbf{w}) - I(\mathbf{x})|^2))}_{Data} + \underbrace{\alpha \Psi(|\nabla u|^2 + |\nabla v|^2)}_{Smoothness} d\mathbf{x} \quad (11)$$

with $\mathbf{x} := (x, y, t)^T$ and $\mathbf{w} := (u, v, 1)^T$. The corresponding Euler-Lagrange equation can be written as

$$\begin{aligned} \Psi'(I_z^2) \cdot (I_x I_z) - \alpha \operatorname{div}(\Psi'(|\nabla u|^2 + |\nabla v|^2) \nabla u) &= 0 \\ \Psi'(I_z^2) \cdot (I_y I_z) - \alpha \operatorname{div}(\Psi'(|\nabla u|^2 + |\nabla v|^2) \nabla v) &= 0 \end{aligned} \quad (12)$$

with

$$\begin{aligned} I_z &= I(\mathbf{x} + \mathbf{u}) - I(\mathbf{x}) \\ I_x &= \frac{\partial I(\mathbf{x} + \mathbf{u})}{\partial x} & I_y &= \frac{\partial I(\mathbf{x} + \mathbf{u})}{\partial y} \end{aligned} \quad (13)$$

where I_z is not a temporal derivative but a difference that is sought to be minimized. Because the energy function is not linear or not even convex, the solution of the Euler-Lagrange equation is quite a challenge. The detailed process of the iteration scheme is presented in [Papenberg et al. 2006]. Here we only show the final linear discretion of the Euler-Lagrange equation for non-linear, non-quadratic data and smoothness terms of the Horn and Schunck model.

Let l denote the iteration index for the outer loop, equivalent to the pyramidal level, k for the inner loop equivalent to the fixed point iteration. The fixed point variables $[du]_{i,j}^{k,l}$ and $[dv]_{i,j}^{k,l}$ are both initialized with 0. Furthermore, let $[\Psi'_D]_{i,j}^{k,l}$ and $[\Psi'_S]_{i,j}^{k,l}$ denote

the robust factor and the diffusivity computed at the fixed point $([du]_{i,j}^{k,l}, [dv]_{i,j}^{k,l})$. Final linear equations at the pixel (i, j) are described as

$$\begin{aligned}
& [\Psi'_D]_{i,j}^{k,l} \cdot ([I_x]_{i,j}^k ([I_z]_{i,j}^k + [I_x]_{i,j}^k [du]_{i,j}^{k,l+1} + [I_y]_{i,j}^k [dv]_{i,j}^{k,l+1})) \\
& -\alpha \sum_{l \in (x,y)} \sum_{(\tilde{i}, \tilde{j}) \in N_l(i,j)} \frac{([\Psi'_S]_{i,j}^{k,l} + [\Psi'_S]_{i,j}^{k,l}) ([u]_{i,j}^k + [du]_{i,j}^{k,l+1}) - ([u]_{i,j}^k + [du]_{i,j}^{k,l+1})}{2 h_l^2} = 0 \\
& [\Psi'_D]_{i,j}^{k,l} \cdot ([I_y]_{i,j}^k ([I_z]_{i,j}^k + [I_x]_{i,j}^k [du]_{i,j}^{k,l+1} + [I_y]_{i,j}^k [dv]_{i,j}^{k,l+1})) \\
& -\alpha \sum_{l \in (x,y)} \sum_{(\tilde{i}, \tilde{j}) \in N_l(i,j)} \frac{([\Psi'_S]_{i,j}^{k,l} + [\Psi'_S]_{i,j}^{k,l}) ([v]_{i,j}^k + [dv]_{i,j}^{k,l+1}) - ([v]_{i,j}^k + [dv]_{i,j}^{k,l+1})}{2 h_l^2} = 0. \quad (14)
\end{aligned}$$

There is a little difference from the non-quadratic robust function, i.e.,

$$\begin{aligned}
[\Psi'_D]_{i,j}^{k,l} &= \frac{1}{2 \sqrt{([I_x]_{i,j}^k [du]_{i,j}^{k,l} + [I_y]_{i,j}^k [dv]_{i,j}^{k,l} + [I_z]_{i,j}^k)^2 + \varepsilon_D^2}} \\
[\Psi'_S]_{i,j} &= \frac{1}{2 \sqrt{|\nabla([u]_{i,j}^k + [du]_{i,j}^{k,l+1})|^2 + |\nabla([v]_{i,j}^k + [dv]_{i,j}^{k,l+1})|^2 + \varepsilon_S^2}}. \quad (15)
\end{aligned}$$

It should be noted that within every pyramidal level, u and v are updated as

$$\begin{aligned}
[u]^{k+1} &= [u]^k + [du]^k \\
[v]^{k+1} &= [v]^k + [dv]^k. \quad (16)
\end{aligned}$$

They stay the same during the fixed iteration inside each pyramidal level. Between levels, bilinear interpolation is used to change the scale of u and v . Discretion yields a linear system of equations, which can be solved by the SOR method [Young 1971].

3.2. Protocol B: Combine Local and Global Model (CLG)

As mentioned before, the Horn and Schunck Model [Horn and Schunck 1981] is the global method that yields dense flow fields, while the Lucas and Kanade Model [Lucas and Kanade 1981] is the local method that is more robust under noise. The idea of these two classic models are both derived from optical flow constraint shown in Equation 3. As mentioned above, this equation itself is not sufficient to compute the flow fields u and v uniquely. In order to deal with this problem, the Lucas and Kanade assumes that the optical flow vector is constant within some neighbourhood with the size ρ . In this case, it is possible to determine the two constants u and v at some location (x, y, t) with a weighted least square fit by minimizing the function

$$E_{LK}(u, v) = \frac{1}{2} K_\rho * (I_x u + I_y v + I_t)^2 \quad (17)$$

where the standard deviation of the Gaussian serves as an integration scale on which the main contribution of the least square fit is computed. Nevertheless, the Horn and Schunck Model assumes that the optical flow vector is globally smooth. Its corresponding energy function is shown in Equation 4. Both local and global methods have their pros and cons. Thus, Bruhn [Bruhn et al. 2005] proposed a new model that combines the robustness of the local method with the density of the global approach. It becomes the famous model called Combining Local and Global (CLG) methods.

3.2.1. B1: Quadratic Data and Smoothness Terms. The quadratic energy function of CLG can be written as

$$E(u, v) = \int_{\Omega} \underbrace{(K_{\rho} * (I_x u + I_y v + I_t))^2}_{Data} + \alpha \underbrace{(|\nabla u|^2 + |\nabla v|^2)}_{Smoothness} dx dy. \quad (18)$$

3.2.2. B2: Non-quadratic Data and Smoothness Terms. In order to make the approach more robust against outliers for both the data and the smoothness term, Bruhn et al [Bruhn et al. 2005] proposed the minimization of the following function that uses a non-quadratic optimization instead of quadratic optimization. It is the same as previous in Horn and Schunck Model, i.e,

$$E(u, v) = \int_{\Omega} \underbrace{(\Psi(K_{\rho} * (I_x u + I_y v + I_t))^2)}_{Data} + \alpha \underbrace{\Psi(|\nabla u|^2 + |\nabla v|^2)}_{Smoothness} dx dy. \quad (19)$$

3.2.3. B3: Non-linear non-quadratic Data and Smoothness Terms. Similarly, Bruhn [Bruhn 2006] proposed the non-linear brightness constancy with the robust non-quadratic function and the weighted least square function to cope with the noise. This optical flow model is described as

$$E(u, v) = \int_{\Omega} \underbrace{(\Psi(K_{\rho} * (|I(\mathbf{x} + \mathbf{w}) - I(\mathbf{x})|^2)))}_{Data} + \alpha \underbrace{\Psi(|\nabla u|^2 + |\nabla v|^2)}_{Smoothness} d\mathbf{x}. \quad (20)$$

The energy function $E(u, v)$ is non-linear and non-convex. Hence the solution obtained by the proposed iteration scheme depends on the initialization heavily. In order to avoid getting trapped in local minima, a coarse-to-fine warping technique was proposed [Brox et al. 2004; Papenberg et al. 2006]. For this purpose, a finest image pyramid is generated that is successively down-sampled by an arbitrary but fixed constant $\eta \in (0, 1)$, usually $\eta = 0.5$ is taken. Choosing larger η results in smoother transitions between levels of pyramids and potentially leads to better results.

3.3. Protocol C: Combine Brightness and Gradient Model (CBG)

The basic assumption of most optic flow methods is the grey value of the shifted pixel stays constant. Unfortunately, it is no longer valid when the brightness in the image sequence changes from one image to the next. In order to deal with non-constant illumination, it is necessary to introduce a constraint that is invariant against brightness changing. Although the gradient can be slightly changed due to changing of the grey value, it is much less dependent on illumination. Hence, one possibility is the assumption that the gradient of the shifted pixel stays constant

$$\nabla I(x, y, t) = \nabla I(x + u, y + v, t + 1). \quad (21)$$

After the Taylor expansion, the constraints can be obtained as follows

$$\begin{aligned} I_{xx}u + I_{xy}v + I_{xt} &= 0 \\ I_{xy}u + I_{yy}v + I_{yt} &= 0 \end{aligned} \quad (22)$$

where the double subscript denotes the second order derivative. In contrast to the brightness constancy assumption requiring the first derivative is not zero, the gradient constancy assumption can only provide information at positions where the second derivative is not zero and it might be more sensitive to noise. Thus, Brox [Brox et al. 2004] proposed to combine the gradient assumption with the brightness assumption with a weighted parameter γ

$$E_{data} = (I_x u + I_y v + I_t)^2 + \gamma((I_{xx}u + I_{xy}v + I_{xt})^2 + (I_{xy}u + I_{yy}v + I_{yt})^2) \quad (23)$$

when γ equals to 0, it becomes the Horn and Schunck Model.

3.3.1. C1: Quadratic Data and Smoothness Terms. According to the new data term under the combined basic smoothness assumption, it is easy to obtain a new energy function as follows

$$E(u, v) = \int_{\Omega} \underbrace{(E_{data})}_{Data} + \alpha \underbrace{(|\nabla u|^2 + |\nabla v|^2)}_{Smoothness} dx dy. \quad (24)$$

3.3.2. C2: Non-quadratic Data and Smoothness Terms. The realization of the non-quadratic penalization is straightforward for single constraint. However, if the data term is based on multiple constraints, the robust non-quadratic function becomes more complicated. The problem occurs particularly when different terms should be penalized. Under this situation, the separate robust strategy of the brightness and the gradient constancy are considered because it makes more sense to make each constancy assumption more robust individually [Bruhn 2006; Zimmer et al. 2011]. So the corresponding energy function can be described as follows:

$$E(u, v) = \int_{\Omega} \underbrace{(\Psi((I_x u + I_y v + I_t)^2) + \gamma \Psi((I_{xx} u + I_{xy} v + I_{xt})^2 + (I_{xy} u + I_{yy} v + I_{yt})^2))}_{Data} + \alpha \underbrace{\Psi(|\nabla u|^2 + |\nabla v|^2)}_{Smoothness} dx dy. \quad (25)$$

3.3.3. C3: Non-linear non-quadratic Data and Smoothness Terms. In order to deal with large displacements, linearisation of the two data terms is strictly avoided. Brox [Brox et al. 2004] proposed a best performance technique that combines these assumptions together: a brightness constancy assumption, a gradient constancy assumption, and a discontinuity-preserving smoothness constraint. A consistent numerical scheme, called the coarse-to-fine warping strategy, based on three nested fixed point iterations was presented [Black and Anandan 1996; Brox et al. 2004; Papenberg et al. 2006]. The corresponding energy function with a non-linear but accurate function in the data term is as follows

$$E(u, v) = \int_{\Omega} \underbrace{(\Psi(|I(\mathbf{x} + \mathbf{w}) - I(\mathbf{x})|^2) + \Psi(|\nabla I(\mathbf{x} + \mathbf{w}) - \nabla I(\mathbf{x})|^2))}_{Data} + \alpha \underbrace{\Psi(|\nabla u|^2 + |\nabla v|^2)}_{Smoothness} d\mathbf{x}. \quad (26)$$

4. ANALYSIS OF COMPUTATIONAL PROPERTIES BASED ON BASIC FUNCTIONAL MODULES

After introducing different methods of variational optical flow estimation, it is necessary to investigate their computational properties to find out factors influencing the real time performance. It will be proceeded at the following steps.

- (1) For clarity of analysing, we extract eight common basic functional modules from nine methods above. Every method can be constituted by some of them, as shown in Table I.
- (2) The computational complexity and parallelizability of each basic functional module are studied.

In following analysis, we assume that the image size is $M \times N$ where M denotes the number of rows and N denotes the number of columns. The sampled image size is $m \times n$ where m denotes the number of rows and n denotes the number of columns.

Table I. Constitution of Different Methods by Basic Functional Modules

| | A1 | A2 | A3 | B1 | B2 | B3 | C1 | C2 | C3 |
|----|----|----|----|----|----|----|----|----|----|
| P1 | ○ | ○ | ✓ | ○ | ○ | ✓ | ○ | ○ | ✓ |
| P2 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| P3 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| P4 | | | | | | | ✓ | ✓ | ✓ |
| P5 | | | | ✓ | ✓ | ✓ | | | |
| P6 | ✓ | | | ✓ | | | ✓ | | |
| P7 | | ✓ | ✓ | | ✓ | ✓ | | ✓ | ✓ |
| P8 | | | ✓ | | | ✓ | | | ✓ |

Note: A1, A2, A3, B1, B2, B3, C1, C2 and C3 denote nine variational optical flow methods above respectively.

P1, P2, P3, P4, P5, P6, P7, and P8 denote eight basic functional modules to constitute those methods above.

“✓” means the module is **required** for some algorithm.

“○” means the module is **optional** for some algorithm.

P1: Image Sampling

P5: Least Square

P2: Pre-smoothing

P6: Linear Solver

P3: Compute 1stDeriv

P7: Non-linear Solver

P4: Compute 2ndDeriv

P8: Image Warping

4.1. Image Sampling (P1)

In Table I, it is easy to find *ImageSampling(P1)* module including down-sampling and over-sampling is a basic process required for methods A3, B3 and C3. The common process of them using this module is the coarse-to-fine technique [Alvarez et al. 2000]. It can not only reduce the computational time for decreasing the image size by down-sampling operation, but also yield better flow fields when combing with warping techniques. Furthermore, as shown in Table I, *ImageSampling(P1)* can also be used in all other methods to obtain better real time performance [Bruhn 2006].

In this section, the relatively simpler method of bilinear interpolation [Gonzalez and Woods 2007] is taken as an instance of *ImageSampling(P1)* module. Figure 1 shows the bilinear interpolation grid. The value of the sub-pixel I is interpolated linearly from its four nearest neighbours (A, B, C, D). First, horizontal interpolated points (E, F) are computed according to the horizontal fractional component $frac(x)$ of the sub-pixel coordinate. Then the final interpolated value is computed from (E, F) according to the vertical fractional component $frac(y)$ of the sub-pixel coordinate.

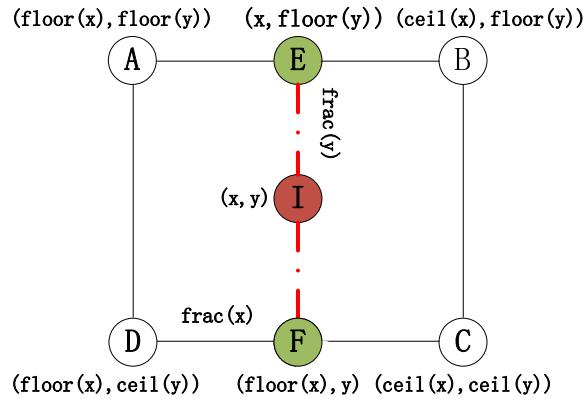


Fig. 1. Bilinear Interpolation Grid.

- *Computational Complexity Analysis.* With the memory chip going larger and cheaper, it is no longer the primary constraint affecting computation. Thus, we will pay more attention to the time complexity of modules in this paper. From Figure 1 and the analysis above, it is easy to know the time complexity for sampling one frame of image is $O(kmn)$, where k is a constant related to the number of neighbours.
- *Parallelizability Analysis.* According to the bilinear interpolation scheme, the value of sub-pixel in the sampling image is computed linearly from its four nearest neighbours in the original images. The interval computational results of four nearest neighbours do not influence each other, they can be computed in parallel. Hence, the module of *ImageSampling(P1)* is straightforward to be implemented in parallel with specific hardware. For instance, Fahmy [Fahmy 2008] introduces a generalised parallel bilinear interpolation architecture for vision systems.

4.2. Pre-smoothing (P2)

As shown in Table I, *Pre-smoothing(P2)* module is required in all variational optical flow methods. It is used to reduce the influence of noise and outliers. It is important because the computation of derivatives relies directly on the smoothed image. For image sequences more than two frames, the spatial-temporal pre-smoothing introduced in [Bruhn 2006] is also a popular mechanism.

The *Pre-smoothing(P2)* module is generally implemented by convolution. For instance, the smoothed image frame $I(x, y, t)$ can be obtained by convolving the original image frame $I_0(x, y, t)$ with a Gaussian kernel K_σ with the standard deviation σ .

$$I(x, y, t) = K_\sigma * I_0(x, y, t). \quad (27)$$

Convolution is a widely used but computationally expensive operation. In order to decrease the computational complexity, two passes of 1D convolution, one in the horizontal and the other in the vertical direction respectively, can be used to replace one pass of 2D convolution under some situation [Naghizadeh and Sacchi 2009]. The acceleration of computational performance by using 1D convolution is shown in Figure 2.

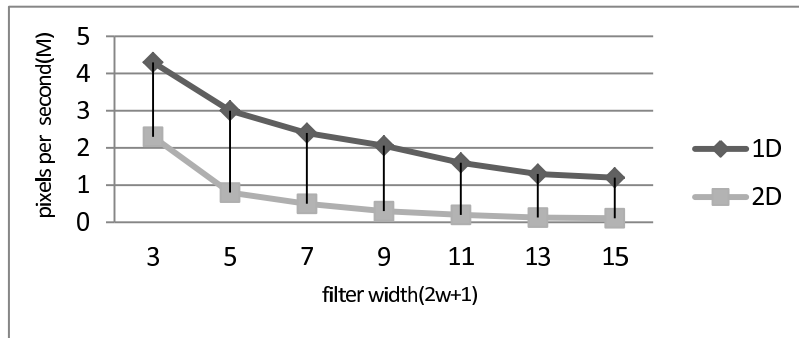
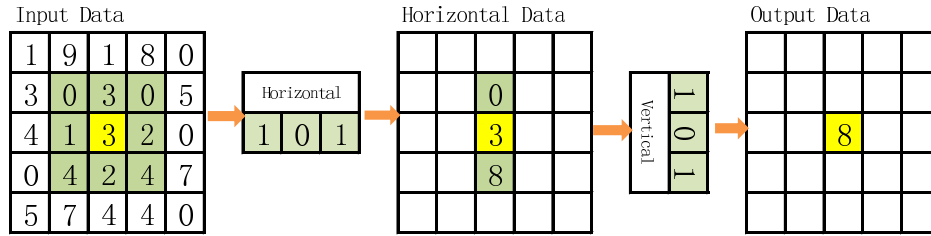


Fig. 2. Performance Comparison between two Separate 1D convolutions and one 2D convolution.

- *Computational Complexity Analysis.* For a given kernel matrix with width w for convolving, it needs w^2MN multiplications for a 2D convolution, while $2wMN$ multiplications for two separate 1D convolution. Furthermore, It still needs $w^2MN - 1$ accumulations for the 2D convolution and $2wMN - 2$ for two 1Ds. Thus, the time complexity for 2D pre-smoothing is roughly $O(2 * w * w * M * N)$, and $O(2 * 2 * w * M * N)$ for 1D pre-smoothing.

- **Parallelizability Analysis.** For 2D convolution, the $w^2 MN$ multiplications are independent and can be computed simultaneously. Thus, they can be computed in real parallel. Furthermore, their w^2 products can be summed together simultaneously if suitable hardware is available [Asano et al. 2009]. Figure 3 shows the process that uses two passes of 1D convolution to replace one 2D convolution. It is obvious that single 1D convolution can be computed in parallel in the similar way with the 2D one. For two passes of 1D convolution, it still can be parallelized by pipelining the two 1D convolution. It shows the time complexity of two passes of 1D convolution can be decreased almost to $O(M * N)$ by implemented in specific hardware architecture [Chai and Shi 2011]. Asano et al show that with the increasing of kernel size, the parallelism of computation will become more critical for real-time computation [Asano et al. 2009].

Fig. 3. 1D Convolution with the kernel size $w = 3$

4.3. Compute 1st Order Derivative (P3)

As shown in Table I, *Compute1stOrderDerivative(P3)* module is required in all variational optical flow methods. The function of this module is to extract the motion tensor from image sequences. It is implemented by 1D convolution with derivative filters on x or y direction, similar to 1D pre-smoothing. For instance, it could simply compute the two-point central difference by convolving with the kernel $[-1, 0, 1]$ in the x , y and t direction respectively. In this paper, we prefer a finite difference scheme involving central differences with fourth-order errors to approximate the spatial derivative at each point. The detailed steps for derivative computing about this scheme are given as follows.

- (1) Averaging the grey value between two consecutive frames of pre-smoothed image sequence, as described in Equation 28 .

$$\tilde{I}(x, y, t) = \frac{I(x, y, t) + I(x, y, t + 1)}{2} \quad (28)$$

- (2) Using the mask $\frac{1}{12}[1, -8, 0, 8, -1]$ [Barron et al. 1994] to compute spatial derivatives for $i = 1, \dots, N$ and $j = 1, \dots, M$ with respect to a rectangular grid with spacing h_x and h_y . It can be written as

$$I_x(i, j, t) = \frac{\tilde{I}(i - 2, j, t) - 8\tilde{I}(i - 1, j, t) + 8\tilde{I}(i + 1, j, t) - \tilde{I}(i + 2, j, t)}{12h_x}$$

$$I_y(i, j, t) = \frac{\tilde{I}(i, j - 2, t) - 8\tilde{I}(i, j - 1, t) + 8\tilde{I}(i, j + 1, t) - \tilde{I}(i, j + 2, t)}{12h_y} \quad (29)$$

- (3) The temporal derivative is computed by a finite difference between two consecutive frames.

$$I_t(i, j, t) = I(i, j, t + 1) - I(i, j, t) \quad (30)$$

It can also try a six-order approximate with the stencil $\frac{1}{60}[-1, 9, -45, 0, 45, -9, 1]$ [Bruhn et al. 2005] to compute spatial derivatives.

- *Computational Complexity Analysis.* It is obvious that all schemes above can be implemented by 1D convolution. Thus the computational complexity can be considered the same with the module of *Pre-smoothing(P2)*.
- *Parallelizability Analysis.* Similarly, the parallelizability of this module is the same with the module of *Pre-smoothing(P2)*.

4.4. Compute 2nd Order Derivative (P4)

As shown in Table I, it is possible to compute higher-order derivatives in some variational methods that are less sensitive to illumination changes. The best idea to compute second derivatives is also to apply the spatial derivative filter based on the information of the first derivatives. The detailed process is described as follows.

- (1) According to the above first derivatives, we already obtained the first derivatives $I_x(i, j, t)$ and $I_y(i, j, t)$. So use the mask $\frac{1}{12}[1, -8, 0, 8, -1]$ to compute second spatial derivatives for $i = 1, \dots, N$ and $j = 1, \dots, M$ with respect to a rectangular grid with spacing h_x and h_y .

$$\begin{aligned} I_{xx}(i, j, t) &= \frac{I_x(i-2, j, t) - 8I_x(i-1, j, t) + 8I_x(i+1, j, t) - I_x(i+2, j, t)}{12h_x} \\ I_{yy}(i, j, t) &= \frac{I_y(i, j-2, t) - 8I_y(i, j-1, t) + 8I_y(i, j+1, t) - I_y(i, j+2, t)}{12h_y} \\ I_{xy}(i, j, t) &= \frac{I_x(i, j-2, t) - 8I_x(i, j-1, t) + 8I_x(i, j+1, t) - I_x(i, j+2, t)}{12h_y} \end{aligned} \quad (31)$$

- (2) In contrast, the second spatial-temporal derivative is computed by a forward difference between two frames combining with a spatial derivative.

$$\begin{aligned} I_{xt}(i, j, t) &= \frac{I_t(i-2, j, t) - 8I_t(i-1, j, t) + 8I_t(i+1, j, t) - I_t(i+2, j, t)}{12h_x} \\ I_{yt}(i, j, t) &= \frac{I_t(i, j-2, t) - 8I_t(i, j-1, t) + 8I_t(i, j+1, t) - I_t(i, j+2, t)}{12h_y} \end{aligned} \quad (32)$$

- *Computational Complexity Analysis.* Obviously, the second spatial-temporal derivative is similar with the module of *Compute1stDerivative(P3)*. It also needs to be implemented by the 1D convolution. Thus the computational complexity can be analysed the same as the module of *Pre-smoothing(P2)*.
- *Parallelizability Analysis.* For the same reason, the parallelizability of this module is the same with the module of *Pre-smoothing(P2)*.

4.5. Least Square (P5)

As shown in Table I, *LeastSquare(P5)* module is required in Prototype B. This module makes the optical flow estimation more robust under noise and considers the neighbourhood information within motion tensors [Lucas and Kanade 1981; Bruhn et al. 2005]. It is commonly used to decrease the weight of neighbouring constraints when it goes farther from the centre, by performing a Gaussian weighted least square fit. That is to say, the motion tensor closer to the neighbourhood's centre would have more influence than farther one.

In general, the strategy of the least square is performed by the 2D convolution with a Gaussian K_ρ having the standard deviation ρ , as shown in Figure 4. Let $J_{new}(\nabla I)$ denote the new motion tensor and $J_{old}(\nabla I)$ denote the original one. The new motion

tensor coped with the least square can be written as

$$J_{new}(\nabla I) = K_{\rho} * J_{old}(\nabla I). \quad (33)$$

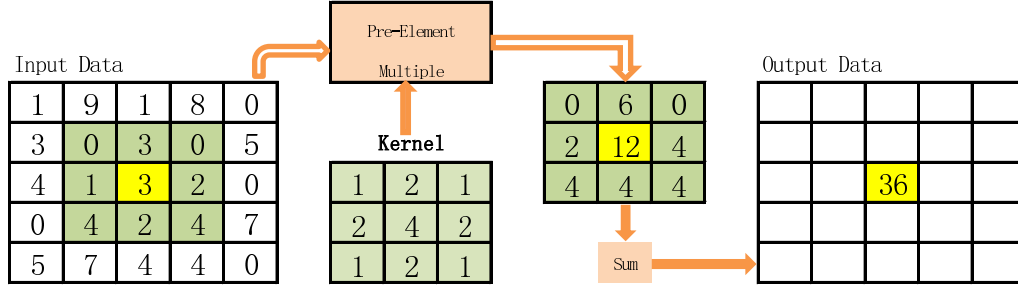


Fig. 4. 2D Convolution using a 3×3 filter

- **Computational Complexity Analysis.** The module of *LeastSquare(P5)* is implemented by the 2D convolution. Thus the computational complexity can be analysed the same way with the 2D *Pre-smoothing(P2)*.
- **Parallelizability Analysis.** The least square is implemented by summing its nine nearest weighted neighbours, just like the operation of 2D convolution. Because of no data dependency existing, the computation of this module is suited to be parallelized. With suitable hardware architecture, nine multiplications can be computed in real parallel, then the summation of their products can also be computed simultaneously. By comparing the performance of the 2D convolution on CPU, GPU and FPGA respectively, it shows that with the kernel size increasing, parallelism becomes more significant for the real-time performance [Asano et al. 2009].

4.6. Linear Solvers (P6)

As shown in Table I, the module of *LinearSolvers(P6)* is used to solve the quadratic terms in variational methods. For simplicity, here we just present two classic methods: Successive Over Relaxation (SOR) and Full Multi-grid (FMG). The SOR method is the best performance of non-hierarchical iterative solvers [Young 1971]. The FMG method can be identified as the most advanced hierarchical strategy [Briggs et al. 2000].

4.6.1. General Problem. As mentioned in section 3, the optical flow problem is actually the problem to solve the large linear equation with the form $Ax = b$, where A is the regular positive semi-definite and symmetric coefficient matrix with size $(2 * M * N) * (2 * M * N)$, x is the variable and b is the constant having the size of the right hand vectors for $(2 * M * N)$ elements each. According to Equation 6, the discrete Euler-Lagrange equation for general linear solvers is described as

$$\begin{aligned}
 [J_{11}]_{i,j} [u]_{i,j} + [J_{12}]_{i,j} [v]_{i,j} + [J_{13}]_{i,j} - \alpha \left(\sum_{l \in (x,y)} \sum_{(\tilde{i}, \tilde{j}) \in N_l(i,j)} \frac{[u]_{\tilde{i}, \tilde{j}} - [u]_{i,j}}{h_l^2} \right) &= 0 \\
 [J_{12}]_{i,j} [u]_{i,j} + [J_{22}]_{i,j} [v]_{i,j} + [J_{23}]_{i,j} - \alpha \left(\sum_{l \in (x,y)} \sum_{(\tilde{i}, \tilde{j}) \in N_l(i,j)} \frac{[v]_{\tilde{i}, \tilde{j}} - [v]_{i,j}}{h_l^2} \right) &= 0 \quad (34)
 \end{aligned}$$

for $i = 1, \dots, N$ and $j = 1, \dots, M$ on a rectangular grid with spacing h_l in l -direction, where $N_l(i, j)$ denotes the set of neighbours of pixel (i, j) in direction of axis l .

A popular iterative technique to solve this linear system starts with an initial approximation to the correct solution, and generates a sequence of vectors that converge to it. These stop criteria can either be constrained by a fixed number of steps based on experiences, or dependent on certain conditions evaluated during iteration.

4.6.2. Linear SOR Solver. The non-hierarchical solver only considers the single grid. Actually, it may take thousands of iterations to transmit local information between unknowns which are not coupled together directly. Jacobi, Gauss-Seidel, and SOR are the three main classic methods based on the splitting-based iteration for non-hierarchical solvers. The SOR solver has the reputation of fast convergence with an optimal computational complexity of $O(n^{1.5})$. It is based on the extrapolation of Gauss-Seidel results [Young 1971]. Let us now study the linear scheme looks like the general problem. The corresponding SOR iteration equations are given by

$$\begin{aligned} [u]_{i,j}^{k+1} &= (1-w)[u]_{i,j}^k + w \frac{\alpha \left(\sum_{(\tilde{i},\tilde{j}) \in N_l^-(i,j)} \frac{[u]_{\tilde{i},\tilde{j}}^{k+1}}{h_l^2} + \sum_{(\tilde{i},\tilde{j}) \in N_l^+(i,j)} \frac{[u]_{\tilde{i},\tilde{j}}^k}{h_l^2} \right) - ([J_{12}]_{i,j}[v]_{i,j}^k + [J_{13}]_{i,j})}{[J_{11}]_{i,j} + \alpha \sum_{(\tilde{i},\tilde{j}) \in N_l(i,j)} \frac{1}{h_l^2}} \\ [v]_{i,j}^{k+1} &= (1-w)[v]_{i,j}^k + w \frac{\alpha \left(\sum_{(\tilde{i},\tilde{j}) \in N_l^-(i,j)} \frac{[v]_{\tilde{i},\tilde{j}}^{k+1}}{h_l^2} + \sum_{(\tilde{i},\tilde{j}) \in N_l^+(i,j)} \frac{[v]_{\tilde{i},\tilde{j}}^k}{h_l^2} \right) - ([J_{12}]_{i,j}[u]_{i,j}^{k+1} + [J_{23}]_{i,j})}{[J_{22}]_{i,j} + \alpha \sum_{(\tilde{i},\tilde{j}) \in N_l(i,j)} \frac{1}{h_l^2}} \end{aligned} \quad (35)$$

for $i = 1, \dots, N$ and $j = 1, \dots, M$. From a computational point of view, the cost to use the SOR scheme is only marginally higher than that of the Gauss-Seidel method. However, one should keep in mind that the SOR method may need much less iterations to converge.

4.6.3. Linear FMG Solver. A basic linear multi-grid strategy to speed up computation is to use the two-grid method [Black and Anandan 1996]. Hereby, we focus on the advanced linear multi-grid solver proposed by [Bruhn and Weickert 2006] for the reason that the FMG solver can accelerate optic flow algorithms significantly even on CPUs.

In order to understand the arithmetic operators involved, let us start by a short introduction of the main strategy for linear FMG solver. This solver is made up of coarse-to-fine technique and linear two-grid algorithms. They make use of sophisticated correction steps which compute the low-frequent errors by classical iterative solvers on a coarser grid. The general linear FMG scheme to solve the equation system can be written as follows

$$A^h x^h = b^h \quad (36)$$

where A^h denotes linear operator and h denotes the grid level. The linear FMG solver is described in Algorithm 1 below [Briggs et al. 2000; Bruhn 2006].

- **Computational Complexity Analysis.** For the linear SOR solver, the computational time is associated with the iteration scheme for convergence. Similar with the other linear non-hierarchical solver, the time complexity of the SOR with better performance is $O((M * N)^{1.5})$ on CPU [Young 1971]. For the linear FMG solver, it can provide better real time performance than other methods showed in [Bruhn 2006], although it may need more memory space to store down-sampled motion tensors. For instance, the time complexity of the full multi-grid is $O((M * N))$ in theory.
- **Parallelizability Analysis.** Both of solvers above may not be suited to be parallelised directly, because of the data dependency existing among pixels within each iteration. So they have to be executed in different order or modified for obtaining better

ALGORITHM 1: Linear FMG Solver

Initial the flow result $x = 0$ on the coarsest grid level ;

for the grid level $h = 1$ step 1 until h equals the finest grid level **do**

 Subroutine L2MG(A, x, b, h) : – Linear Two-grid Function

begin of L2MG

if h eq 1 **then**

 SOR($A, x, b, 1, h$); – Smoothing Relaxation

else

 SOR($A, x, b, 2, h$); – Pre-Smoothing Relaxation

$r^h = b^h - A^h x^h$; – Residual Equation Computation

$b^{h-1} = R^{h-1} r^h$; – Restriction Operation

$x^{h-1} = 0$; – Initial Coarse Solution

 L2MG($A, x, b, h-1$); – Coarse Grid Computation

$x^h = x^h + P^h x^{h-1}$; – Fine Grid Correction

 SOR($A, x, b, 2, h$); – Post-Smoothing Relaxation

end

end of L2MG

if h is less than the finest grid level **then**

 Update the new flow result x by applying prolongation operation to the finer grid;

else

 break;

end

end

Note: R denotes Restriction Operator and P denotes Prolongation Operator.

For SOR($A, x, b, 2, h$), 2 denotes the relaxation iterations.

parallelism. Presently, Gwosdek et al [Gwosdek et al. 2010] propose a red-black relaxation more suited to be processed on parallel processors. Similarly, advanced hierarchical solvers have also been improved for being processed in parallel, such as in [Grossauer and Thoman 2008; Sundberg et al. 2011].

4.7. Non-linear Solver (P7)

After showing how to design numerical schemes for the linear case, the systematic approach for the non-linear case will be introduced in this section. As shown in Table I, the non-linear solver is used in non-quadratic and non-linear terms.

4.7.1. General Problem. Let us first describe the general problem of the non-linear equation system as the same way with the linear case. According to Equation 8 and 14, this can be written as

$$\begin{aligned}
 & [\Psi'_D]_{i,j}^k [J_{11}]_{i,j} [u]_{i,j}^{k+1} + [\Psi'_D]_{i,j}^k [J_{12}]_{i,j} [v]_{i,j}^{k+1} + [\Psi'_D]_{i,j}^k [J_{13}]_{i,j} \\
 & - \alpha \sum_{l \in (x,y)} \sum_{(\tilde{i}, \tilde{j}) \in N_l(i,j)} \frac{([\Psi'_S]_{\tilde{i}, \tilde{j}}^k + [\Psi'_S]_{i,j}^k) ([u]_{\tilde{i}, \tilde{j}}^{k+1} - [u]_{i,j}^{k+1})}{2 h_l^2} = 0 \\
 & [\Psi'_D]_{i,j}^k [J_{12}]_{i,j} [u]_{i,j}^{k+1} + [\Psi'_D]_{i,j}^k [J_{22}]_{i,j} [v]_{i,j}^{k+1} + [\Psi'_D]_{i,j}^k [J_{23}]_{i,j} \\
 & - \alpha \sum_{l \in (x,y)} \sum_{(\tilde{i}, \tilde{j}) \in N_l(i,j)} \frac{([\Psi'_S]_{\tilde{i}, \tilde{j}}^k + [\Psi'_S]_{i,j}^k) ([v]_{\tilde{i}, \tilde{j}}^{k+1} - [v]_{i,j}^{k+1})}{2 h_l^2} = 0
 \end{aligned} \tag{37}$$

where non-linear abbreviations $[\Psi'_D]_{i,j}$ and $[\Psi'_S]_{i,j}$ denote functions $\Psi'_D(s^2)$ and $\Psi'_S(s^2)$ evaluated at pixel (i, j) . It should be noted that they are evaluated at the previous step k instead of $k + 1$.

4.7.2. Non-linear SOR Solver. Popular approaches to solve the non-linear system are mainly nested iterations in non-hierarchical numerics. Here we focus on the SOR method with coupled point relaxation (CPR) [Bruhn 2006]. The detailed steps to solve the basic non-linear case can be described as

- (1) In order to remove non-linearity, $[\Psi'_D]_{i,j}$ and $[\Psi'_S]_{i,j}$ are fixed to an initial solution (u, v) first.
- (2) Solve resulted linear system to obtain a new fixed point similar with the linear solver.
- (3) Update $[\Psi'_D]_{i,j}$ and $[\Psi'_S]_{i,j}$ for each outer iteration, then GOTO step 2.

4.7.3. Non-linear FMG Solver. For equations of type $A^h(x^h) = b^h$, it is necessary to find a suitable coarse grid representation for the initial problem because the non-linear operator $A^h(x^h)$ needs to be evaluated. The non-linear FMG solver is also made up of the coarse-to-fine technique and two-grid method. As shown in Algorithm 2, it needs to restrict the preliminary solution \tilde{x}^h and use it to evaluate the non-linear operator $A^{h-1}(\tilde{x}^{h-1})$ in non-linear case.

ALGORITHM 2: Non-linear FMG Solver

Initial the flow result $x = 0$ and initial guess flow $\tilde{x} = 0$ on the coarsest grid level ;
for the grid level $h = 1$ step 1 until h equals the finest grid level **do**
 Subroutine N2MG(A, \tilde{x}, x, b, h) : – Non-linear Two-grid Function
 begin of N2MG
 if h eq 1 **then**
 NSOR($A, \tilde{x}, x, b, 1, h$); – Smoothing Relaxation
 else
 NSOR($A, \tilde{x}, x, b, 2, h$); – Pre-Smoothing Relaxation
 $r^h = b^h - A^h x^h$; – Residual Equation Computation
 $\tilde{x}^{h-1} = R^{h-1} x^h$; – Guess Flow Restriction Operation
 $b^{h-1} = A^{h-1}(\tilde{x}^{h-1}) + R^{h-1} r^h$; – Modify Right Hand Side
 N2MG($A, \tilde{x}, x, b, h-1$); – Coarse Grid Computation
 $x^h = x^h + P^h(x^{h-1} - \tilde{x}^{h-1})$; – Fine Grid Correction
 NSOR($A, \tilde{x}, x, b, 2, h$); – Post-Smoothing Relaxation
 end
 end of N2MG
 if h is less than the finest grid level **then**
 Update the new flow result x by applying prolongation operation to the finer grid;
 Copy the corresponding flow result to the guess flow \tilde{x} ;
 else
 break;
 end
end
Note: R denotes Restriction Operator and P denotes Prolongation Operator.
 For NSOR($A, \tilde{x}, x, b, 2, h$), 2 denotes the relaxation iterations.

- **Computational Complexity Analysis.** In contrast to Linear Solvers (P5), the updated non-linearity is the additional part in non-hierarchical solvers. Thus the corresponding time complexity of this module is $O(i * (M * N)^{1.5})$, where i denotes the iteration index for removing the non-linear. Similarly, the time complexity of the non-linear full multi-grid is $O(i * M * N)$.
- **Parallelizability Analysis.** Similar to the linear case, as the flow field result of each pixel depends on the neighbouring pixel's result within one iteration, it must be

modified for being computed in parallel [Grossauer and Thoman 2008; Sundberg et al. 2011].

4.8. Image Warping (P8)

As shown in Table I, *ImageWarping(P8)* is usually combined with the coarse-to-fine technique. It is a popular tool to improve the performance of optical flow methods [Black and Anandan 1996; Brox et al. 2004; Papenberg et al. 2006; Brox and Malik 2011]. This module reduces the displacement between two frames to keep the constancy assumption available. The detailed steps of *ImageWarping(P8)* is proceeded as follows

- (1) Compute expressions of type $I(\mathbf{x} + \mathbf{w})$ for given flow \mathbf{w} .
 - (2) The second frame of image at $t + 1$ is warped by means of the interpolation (u, v) .
 - (3) Compute $I_{new}(x) = I(x + w) = I(x + u, y + v, t + 1)$.
- *Computational Complexity Analysis.* The bilinear interpolation is used to get the warped image. So the time complexity is $O(M * N)$ that is similar with *ImageSampling(P1)*.
 - *Parallelizability Analysis.* For the same reason, the parallelizability of this module is the same as *ImageSampling(P1)*.

5. EXPERIMENTAL RESULTS AND ANALYSIS

In section 4, the time complexity and the parallelizability of eight functional modules being used to compose different variational methods are discussed theoretically. In order to make the analysis above more intuitive to understand, experimental results and corresponding empirical analysis will be given in this section. Firstly, the percentage of time consumption of each functional module in different methods is displayed. Thus, it is more intuitive to know which modules are key factors affecting the real-time performance of the system. Secondly, concrete numbers of the execution time for all functional modules are listed. Thus, we can have a rough idea about the execution time based on a general-purposed computing platform. Thirdly, the decrease of execution time for the most time-consuming module (P6 or P7) accelerated by kind of parallelism is shown. It shows the direction to make our effort to when the real-time performance is required. Fourthly, the execution time and the AAE (average angular error) [Barron et al. 1994; Baker et al. 2011] of different methods with SOR and FMG schemes respectively are displayed. By combining the most important two factors together, one can learn the properties of different methods clearly by comparison. Finally, the flow field results of different methods with SOR and FMG schemes respectively are displayed in appendix for referencing to get the intuitive idea.

By combining the theoretical analysis with the empirical experiment, it will be helpful for users to make a better trade-off between the flow field results and the real-time performance by selecting different functional modules under specific situations. Furthermore, it is significant for users to know which modules should be selected first to improve their real-time performance.

In this section, all computation are carried out on a desktop computer with a 1.50 GHz Intel Core(TM)2 Duo CPU, 2GB memory space. All algorithms are implemented in ANSI C instead of Matlab code to obtain the execution time more accurately. The source code of all algorithms discussed in this paper can be downloaded from <http://isnc.jiangnan.edu.cn/EmdLab/OpticalFlow.html>. The image sequence for testing is the famous Yosemite sequence with clouds obtained from <ftp://csd.uwo.ca/pub/vision>. The size of Yosemite image sequences is 316×252 . Furthermore, the filter size for Pre-smoothing is 11, for Derivative is 5 and for Least Square is 5×5 . The down-sampling is 0.8 in A3, B3 and C3. Of course, we can select

a more advanced computer and image sequences with higher resolution to get more persuasive results. However, experiments performed here has been already enough for users to know key factors effecting the real-time performance of the optical flow computation.

5.1. Relative and Absolute Time Consumption of Each Module in different methods

Figure 5 and Figure 6 show the percentage of time consumption of each basic functional module in nine variational methods, based on the SOR and the FMG solver respectively. Table II shows the absolute execution time of each module in nine variational methods.

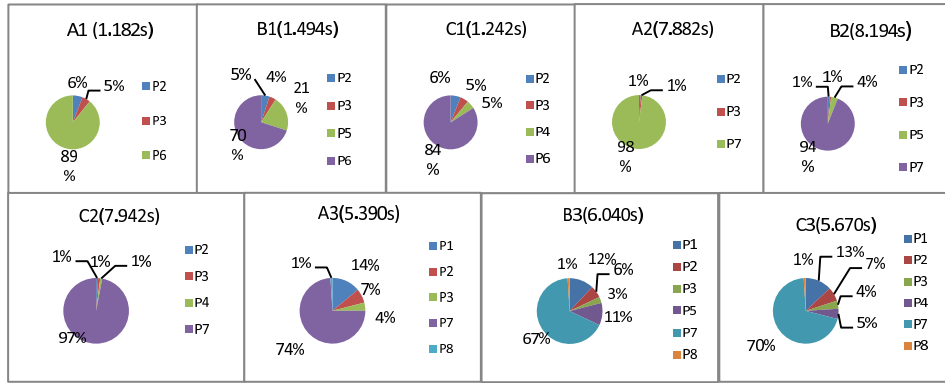


Fig. 5. Percentage of Module's Time Consumption based on the SOR Solver. $P1 - P8$ and $A1 - C3$ are the same as shown in Table I.

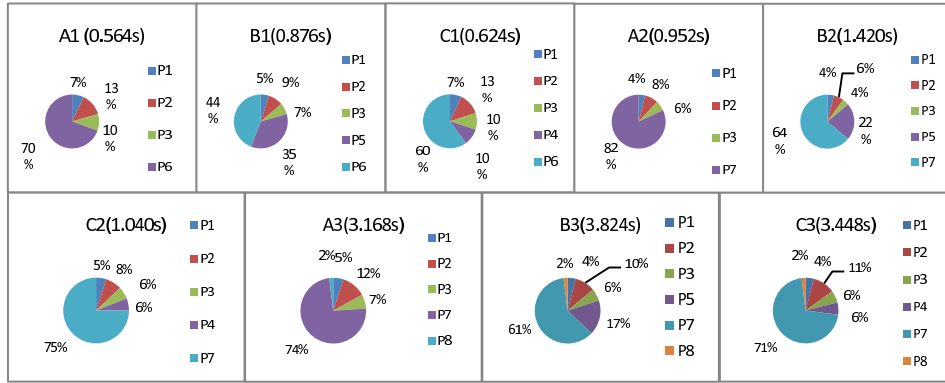


Fig. 6. Percentage of Module's Time Consumption based on the FMG Solver. $P1 - P8$ and $A1 - C3$ are the same as shown in Table I.

As shown in Figure 5, $LinearSolver(P6)$ and $Non-linearSolver(P7)$ consume the largest portion of time in all methods, even more than ninety percent in some methods. Thus, they are the modules should be accelerated first. From Figure 5 to Figure 6, the relative percentage of time consumption in each method changes a lot when transferring from the SOR solver to that of the FMG. By investigating Table II, we can see the reason of changing is primarily due to the time change caused by $P6$ and $P7$, while the time of other modules keeps almost unchanged. Thus, it is obvious the

Table II. Absolute Time Performance of Each Module

| Modules | Common | | | | | SOR | | | FMG | | |
|------------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| Methods[s] | P2 | P3 | P4 | P5 | P8 | P1 | P6 | P7 | P1 | P6 | P7 |
| A1 | 0.076 | 0.056 | | | | | 1.050 | | 0.040 | 0.392 | |
| A2 | 0.076 | 0.056 | | | | | | 7.750 | 0.040 | | 0.780 |
| A3 | 0.380 | 0.200 | | | 0.060 | 0.750 | | 4.000 | 0.056 | | 2.400 |
| B1 | 0.076 | 0.056 | | 0.312 | | | 1.050 | | 0.040 | 0.392 | |
| B2 | 0.076 | 0.056 | | 0.312 | 0.060 | | | 7.750 | 0.040 | | 0.780 |
| B3 | 0.380 | 0.200 | | 0.650 | | 0.750 | | 4.000 | 0.056 | | 2.400 |
| C1 | 0.076 | 0.056 | 0.060 | | | | 1.050 | | 0.040 | 0.392 | |
| C2 | 0.076 | 0.056 | 0.060 | | | | | 7.750 | 0.040 | | 0.780 |
| C3 | 0.380 | 0.200 | 0.280 | | 0.060 | 0.750 | | 4.000 | 0.056 | | 2.400 |

Note: P1 – P8 and A1 – C3 are the same as shown in Table I.

hierarchical scheme is one of the efficient way to improve the real-time performance of the optical flow estimation system.

However, even accelerated by the hierarchical scheme, P6 and P7 still occupies the largest portion of time in all methods as shown in Figure 6. Moreover, according to the concrete numbers, the real-time performance of the whole method still has to be improved further for real applications. Fortunately, as discussed in section 4.6 and 4.7, this module is capable to be parallelised, and some work has been introduced already [Grossauer and Thoman 2008; Sundberg et al. 2011]. Figure 7 shows improved results of methods with P6 accelerated by parallelism. Reassuringly, it is obvious that the execution time of P6 decreases dramatically. It is not any more the largest part of the method. The execution time of whole methods decreases accordingly.

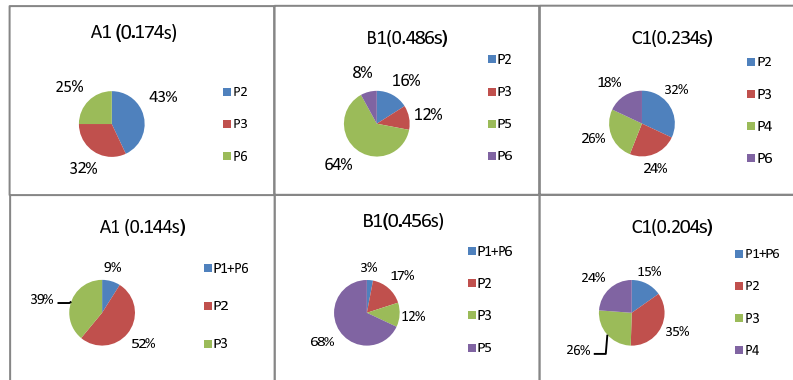


Fig. 7. Percentage of Module's Time Consumption of P6 on Parallel Processor in A1, B1 and C1. (1)Top Row: The SOR Solver. (2)Bottom Row: The FMG Solver.

Although the real-time performance of methods above after being accelerated by parallelism might still not meet the time constraint for real world applications, it really gives a promising direction to improve the real-time performance of optical flow estimation. For instance, P2 and P5 implemented by the 1D or 2D convolution spend most of the time as shown in Figure 7, but they are easy to be accelerated by parallelism as introduced in section 4.

As introduced above, there are some potential approaches to improve the real-time performance of optical flow estimation, such as the FMG scheme, hardware parallelism and so on. Furthermore, the filter size of convolution used in P2, P3, P4 and P5 is also an important factor to influence the real-time performance. All of these factors need to be considered carefully when implementing a specific optical flow estimation system.

5.2. Considering Time Performance and AAE together in Each Method

After discussing the time performance and the potential improvement from the aspect of modules, the time performance and the AAE will be considered together and discussed from the aspect of whole methods. Users should take both factors into account when implementing a specific optical flow system. The discussion is carried out based on the SOR and the FMG scheme respectively. Because it means a better optical flow system that can compute the optical flow field with a lower AAE in a shorter time, we can simply represent the Overall Result (OR) of a optical flow system by the product of the execution time and the AAE (i.e., time*AAE). The smaller one denotes the better system. Surprisingly, as found in Table III, all the three methods with quadratic terms has the best comprehensive result of the time performance and the AAE, for both SOR and FMG schemes. However, for the SOR scheme, the three methods with non-quadratic terms perform worst. For the FMG scheme, the three with both non-quadratic and non-linear terms perform worst. Obviously, the FMG scheme is used primarily for decreasing the execution time, and it is most efficient for the methods with non-quadratic terms. Of course, discussion above is just a rough estimation. For real applications, users should make the trade-off more carefully based on the time performance or the AAE required. Furthermore, the number of iterations is also a key factor to influence the time performance and the AAE.

Please note that, for the SOR scheme, 100 denotes 100 inner SOR iterations. 10 – 50 denotes 10 inner iterations with 50 fixed point iterations. 10 – 5 – 20 denotes 10 outer iterations and 5 inner iterations with 20 fixed point iterations for each outer iterations.

For the FMG scheme, $L - W$ denotes linear full multi-grid with 2 W-cycles with 2 pre-smoothing and post-smoothing relaxation iterations. $N - W$ denotes non-linear full multi-grid with 2 W-cycles with 2 pre-smoothing and post-smoothing relaxation iterations. $10 - N - W$ denotes 10 outer iterations with non-linear full multi-grid solver.

Table III. Time Performance and AAE of Each Method

| Solvers | SOR | | | | FMG | | | |
|---------|-------------|----------|-------|--------|--------------|----------|-------|--------|
| Methods | Iterations | Time [s] | AAE | OR | Iterations | Time [s] | AAE | OR |
| A1 | 100 | 1.182 | 8.71° | 10.295 | $L - W$ | 0.564 | 8.95° | 5.048 |
| B1 | 100 | 1.494 | 8.68° | 12.968 | $L - W$ | 0.876 | 8.75° | 7.665 |
| C1 | 100 | 1.242 | 7.62° | 9.464 | $L - W$ | 0.624 | 7.36° | 4.593 |
| A2 | 10 – 50 | 7.882 | 8.67° | 68.337 | $N - W$ | 0.952 | 8.53° | 8.121 |
| B2 | 10 – 50 | 8.194 | 7.73° | 63.340 | $N - W$ | 1.420 | 6.78° | 9.628 |
| C2 | 10 – 50 | 7.942 | 3.53° | 28.035 | $N - W$ | 1.040 | 4.00° | 4.160 |
| A3 | 10 – 5 – 20 | 5.390 | 7.96° | 42.904 | 10 – $N - W$ | 3.168 | 8.00° | 25.344 |
| B3 | 10 – 5 – 20 | 6.040 | 7.49° | 45.240 | 10 – $N - W$ | 3.824 | 7.76° | 29.674 |
| C3 | 10 – 5 – 20 | 5.670 | 3.33° | 18.881 | 10 – $N - W$ | 3.448 | 3.30° | 11.378 |

Note: P1 – P8 and A1 – C3 are the same as shown in Table I.

AAE: Average Angular Error OR: Overall Results (time*AAE)

6. CONCLUSIONS

The real-time performance is the key factor for the optical flow computing system to be used in real world applications, particularly in embedded systems. In this paper, a systematic analysis about the real-time performance of variational optical flow methods is introduced. According to the analysis above, we can draw conclusions as follows. The FMG scheme is an efficient mechanism to improve the real-time performance of optical flow methods. Methods with quadratic terms generally perform better instead of methods with more advanced terms as we expected. Parallelism is another efficient way to improve the real-time performance of variational optical flow methods.

There are still much work need to do for improving the real-time performance of optical flow methods. It is because even the image sequence with a very small size cannot be processed in real-time yet as shown in section 5. It becomes worse when the increasing requirement for image/video sequences with higher frame rate and higher resolution is taken into account. Thus, these potential approaches for this purpose should be considered. Firstly, algorithms with better parallelizability should be proposed. So they are possible to be computed in real parallel by FPGAs, GPUs, multi-cores and other parallel computing platforms. Secondly, suitable hardware architectures for some functional module should be investigated. For instance, hardware acceleration for P7 can be implemented in addition to P6 proposed in [Grossauer and Thoman 2008; Sundberg et al. 2011]. Thirdly, the hardware architecture for the whole method should also be considered. Because the pipelining technique can be deployed for the whole method to improve the real-time performance further without requiring single module modified. Furthermore, the more general architecture suitable for optical flow computation or the reconfiguration mechanism to meet different methods should be established.

APPENDIX

For intuitively comparison, corresponding flow fields of nine variational optical flow methods with the SOR and the FMG scheme respectively are shown in Figure 8 below.




















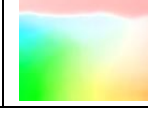
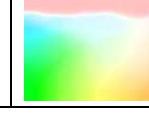
| | Frame 7 of Yosemite with Clouds | | | Frame 8 of Yosemite with Clouds | | | Ground truth(color plot) | | |
|---|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|--|--|
| |  | | |  | | |  | | |
| D | | | | | | | | | |
| S | SOR | | | FMG | | | | | |
| M | 1 | 2 | 3 | 1 | 2 | 3 | | | |
| A |  |  |  |  |  |  | | | |
| B |  |  |  |  |  |  | | | |
| C |  |  |  |  |  |  | | | |

Fig. 8. Performance of Colour Flow Fields
Note: D:Image Data S:Solvers M:Methods

REFERENCES

ALVAREZ, L., WEICKERT, J., AND SANCHEZ, J. 2000. Reliable estimation of dense optical flow fields with large displacements. *International Journal of Computer Vision* 39, 1, 41–56.

- ASANO, S., MARUYAMA, T., AND YAMAGUCHI, Y. 2009. Performance comparison of fpga,gpu and cpu in image processing. *2009 International Conference on Field Programmable Logic and Applications*.
- BAKER, S., SCHARSTEIN, D., LEWIS, J. P., ROTH, S., BLACK, M. J., AND SZELISKI, R. 2011. A database and evaluation methodology for optical flow. *International Journal of Computer Vision* 92, 1–31.
- BARRON, J. L., FLEET, D. J., AND BEAUCHEMIN, S. S. 1994. Performance of optical flow techniques. *International Journal of Computer Vision* 12, 1, 43–77.
- BEAUCHEMIN, S. S. AND BARRON, J. 1995. The computation of optical flow. *ACM Computing Surveys* 27, 3.
- BLACK, M. J. AND ANANDAN, P. 1996. The robust estimation of multiple motions: Parametric and piecewise smooth flow fields. *Computer Vision and Image Understanding* 63, 1, 45–104.
- BRIGGS, W. L., HENSON, V. E., AND MCCORMICK, S. F. 2000. A multigrid tutorial. SIAM, Philadelphia.
- BROX, T. 2005. From pixels to regions: partial differential equations in image analysis. Ph.D. thesis, Saarland University, Saarbrücken, Germany.
- BROX, T., BRUHN, A., PAPENBERG, N., AND WEICKERT, J. 2004. High accuracy optic ow estimation based on a theory for warping. *In Proc. 8th European Conference on Computer Vision* 4, 25–36.
- BROX, T. AND MALIK, J. 2011. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 3, 500–513.
- BRUHN, A. 2006. Variational optic flow computation - accurate modelling and efficient numerics. Ph.D. thesis, Saarland University, Saarbrücken, Germany.
- BRUHN, A. AND WEICKERT, J. 2006. A multigrid platform for real-time motion computation with discontinuity-preserving variational methods. *International Journal of Computer Vision* 70, 257–277.
- BRUHN, A., WEICKERT, J., AND SCHNORR, C. 2005. Lucas/kanade meets horn/schunck: Combining local and global optic ow methods. *International Journal of Computer Vision* 61, 3, 211–231.
- CHAI, Z. AND SHI, J. 2011. Improving klt in embedded systems by processing oversampling video sequence in real-time. *2011 International Conference on Reconfigurable Computing and FPGAs*.
- FAHMY, S. A. 2008. Generalised parallel bilinear interpolation architecture for vision systems. *2008 International Conference on Reconfigurable Computing and FPGAs*.
- GONZALEZ, R. AND WOODS, R. 2007. Digital image processing, 3rd edition. Prentice Hall, Upper Saddle River, NJ 07458.
- GROSSAUER, H. AND THOMAN, P. 2008. Gpu-based multi-grid:real-time performance in high resolution nonlinear image proccsing. *ICVS 2008*, 141–150.
- GWOSDEK, P., BRUHN, A., AND WEICKERT, J. 2010. Variational optic ow on the sony playstation 3. *Real-Time Image Proc (2010)* 5.
- HORN, B. K. P. AND SCHUNCK, B. G. 1981. Determining optical flow. *Artificial Intelligence* 17, 185–203.
- LUCAS, B. D. AND KANADE, T. 1981. An iterative image registration technique with an application to stereo vision. *Proc 7th Intl Joint Conf on Artificial Intelligence(IJCAI)*, 674–679.
- MEMIN, E. AND PEREZ, P. 1998. A multigrid approach for hierarchical motion estimation. *In Proc. 6th International Conference on Computer Vision*.
- NAGEL, H.-H. AND ENKELMANN, W. 1986. An investigation of smoothness constraints for the estimation of displacement vector elds from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8, 5.
- NAGHIZADEH, M. AND SACCHI, M. D. 2009. Multidimensional convolution via a 1d convolution algorithm. *The Leading Edge*.
- PAPENBERG, N., BRUHN, A., BROX, T., DIDAS, S., AND WEICKERT, J. 2006. Highly accurate optic flow computation with theoretically justied warping. *International Journal of Computer Vision* 67, 2, 141–158.
- SUNDARAM, N., BROX, T., AND KEUTZER, K. 2010. Dense point trajectories by gpu-accelerated large displacement optical flow. Tech. Rep. EECS-2010-104, Electrical Engineering and Computer Sciences, University of California at Berkeley. <http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-104.html>.
- SUNDBERG, P., BROX, T., MAIRE, M., ARBELAEZ, P., AND MALIK, J. 2011. Occlusion boundary detection and figure/ground assignment from optical flow. *The 24th IEEE Conference on Computer Vision and Pattern Recognition*.
- WEDEL, A., POCK, T., ZACH, C., BISCHOF, H., AND CREMERS, D. 2008. An improved algorithm for tv-l1 optical flow. *In Proceedings of the Dagstuhl motion workshop*.
- WEICKERT, J. AND SCHNORR, C. 2001. A theoretical framework for convex regularizers in pde-based computation of image motion. *International Journal of Computer Vision* 45, 3, 245–264.
- YOUNG, D. M. 1971. Iterative solution of large linear systems. Academic Press, New York, NY.

ZIMMER, H., BRUHN, A., AND WEICKERT, J. 2011. Optic flow in harmony. *Int J Comput Vis.*