MOTION PARAMETER ESTIMATION BASED ON THE BLOCK RECURSIVE ALGORITHM WITH FINITE WORD LENGTH

Pei-Chuan Liu, Wen-Thong Chang and Wen-Zen Shen
Department of Electronic Engineering
Chiao-Tung University, Hsinchu 30050
Taiwan, R.O.C.

Abstract

Three parameter motion model can better describe the motion trajectory of picture element in video sequence. Combining the features of block process and recursive estimation, a block recursive algorithm is proposd to calculate these three motion parameters with compensating the variety of block position to get better stability. To get the simplification requirement for hardware implementation, the representation accuracy of displaced frame difference and gradient that are given is analyzed. From these analysis we propose a quantized block recursive algorithm with quantized grandient to reduce the computational complexity. With this simplification the quantized motion estimation algorithm could get sufficient computational speed to meet the requirement of real time application with the least accuracy loss.

I. Introduction

In the past 2 parameter motion model for video coding based on the assumption of pure 2-D translatory moving object is widely used. But for the actual image signal with 3-D motion, this assumption may be invalid. Recently, a three parameter motion model has been actively studied [1]. As Hotters[1] proposed, one parameter describes the change in focal length corresponding to the ratio of the focal lengthes before and after zooming, the other two parameters are used to describe the pan or translatory motion of camera. In order to represent the more complex motion of object, some other motion models with more motion parameters have been proposed [2], such as the 6 parameter affine transform, the 8 parameter bilinear transform and the 8 parameter perspective transform, etc. But in the real world such a special motion of object does not usually exist. So most of the motion situation in the real image sequence can be well described with the three parameter motion model.

To calculate the motion parameters of moving object, most of the motion compensation methods belong to two categories : block matching algorithm[3] and the pel-recursive algorithm[4]. Because of the warping coordinate of pixels, the block matching algorithm can't be used to calculate the three motion parameters in three parameter motion model. So the pel-recursive algorithm is the only method that can be used for the motion estimation in three parameter motion model. But for motion estimation block process can get the more stable result than single pixel process. Using the block processing technique to estimate the coefficients of filter based on the LMS algorithm, Clark et al. proposed the block least mean square (BLMS) algorithm [5] and get better performance. It combines the well known sum square error and the mean square error to form the block mean square error(BMSE) performance measure. Combining the block processing operation of block matching method and the iterative searching procedure of pel-recursive algorithm, a block recursive algorithm is proposed to calculate the motion parameters of object. This method is proven to get the better performance than that of the pel recurive method. According to the equation derived in our proposed algorithm we find the block position is an important criterion to influence the stability of the proposed algorithm. Some big coordinate of pixels in a block will influence the estimation algorithm to be diverged. To solve the problem of unstability, some modifications are made for the block recursive algorithm. With the modification on the convergence factor the stability of the proposed algorithm is well improved.

In spite of the improvement on stability, the block recursive algorithm still meets the problem of large amount of calculations. This problem makes the proposed algorithm hard to be implemented and used for real time calculation. For this, some simplifications must be used.

Finite word length is a important criterion in solving the problem of digital signal processing. As compared to the infinite word length of analog signal, the finite word length of digital signal get the less representation range and the worse data accuracy. But on the concept of transmission or calculation, the finite word length of digital signal get the best quality of precision and the high speed of transmitting or calculating rate. Recently quantized state adaptive algorithms [6-7] have been used to reduce the numerical complexity and dynamic range of the least mean square adaption by replacing multiplifications with shifts, bit comparisons or table lookup. Using the concept of quantization, we investigate the influences caused by changing the word length on different calculation parameters and propose a simplified block recursive algorithm to improve the computational speed with the least performance loss. The trade for good performance on computational efficiency is the worse convergence speed. The rest of this paper is organized as follows. In section II we first briefly introduce the three parameter motion model and then propose a block recursive algorithm to calculate these parameters. For the better stability some modifications are made for the block recursive algorithm. For simplification, the influence of finite word length on different calculation

Contributed Paper
Manuscript received July 21, 1997

parameters is analyzed and a quantized block recursive algorithm is proposed in section III. The multiplication reduction according to the coordinate relationship is also presented in this section. Finally, a conclusion is made in section IV.

II The block recursive algorithm

2.1 the three parameter motion model

The three parameter motion model can be described as the transformation of the coordinate

$$(X_2,Y_2) = A(X_1,Y_1) = (a_1X_1+a_2\ ,\ a_1Y_1+a_3) \qquad (1)$$

where (X_1,Y_1) and (X_2,Y_2) are the coordinates of an image point before and after movement in the image space. Parameters a_1, a_2 and a_3 are the three motion parameters that are to be estimated. When parameter $a_1 = 1$, the three parameter model will degenerate to the traditional translational two parameter motion model with $a_2 = dx$ and $a_3 = dy$. To derive the estimation algorithm, the general relationship between the image point movement and the object motion in real world is first reviewd. For object motion, assuming that a particular point (x_1,y_1, z_1) has moved to a new position (x_2,y_2,z_2) with

$$\begin{aligned} x_2&=x_1+dx,\\ y_2&=y_1+dy,\\ z_2&=z_1+dz, \end{aligned} \qquad (2)$$

where (dx,dy,dz) is the 3-D motion vector in the object space. According to the camera projection rule, in the image space, we have

$$\begin{aligned} X_1&=F\bullet x_1/z_1 \qquad & Y_1&=F\bullet y_1/z_1\\ \text{and}&\\ X_2&=F\bullet x_2/z_2 \qquad & Y_2&=F\bullet y_2/z_2 \end{aligned} \qquad (3)$$

From Eqn.(2) and Eqn.(3), the relationship between (X_1, Y_1) and (X_2, Y_2) is

$$\begin{aligned} X_2 &= F\bullet x_2/z_2 = F\bullet(x_1+dx)/(z_1+dz) = F\bullet(x_1+dx)/(z_1(1+k))\\ &= F/(1+k)\bullet x_1/z_1 + F/(1+k)\bullet dx/z_1\\ &= a_1X_1 + a_2 \end{aligned}$$

Similarly,

$$Y_2 = a_1Y_1 + a_3 \qquad (4)$$

with

$$k = dz/z_1$$

$$a_1 = 1/(1+k)$$

$$a_2 = \frac{F}{(1+k)}\times\frac{dx}{z_1}$$

$$a_3 = \frac{F}{(1+k)}\times\frac{dy}{z_1}$$

For camera zooming, the focal length will change from F_1 to F_2. In this case

$$\begin{aligned} X_1&=F_1\bullet x_1/z_1 \qquad & Y_1&=F_1\bullet y_1/z_1\\ \text{and}&\\ X_2&=F_2\bullet x_1/z_1 \qquad & Y_2&=F_2\bullet y_1/z_1 \end{aligned}$$

and the motion parameters will be,

$$a_1=F_2/F_1 \qquad a_2 = a_3 = 0 \qquad (5)$$

2.2 The block recursive algorithm

Based on the Taylor series expansion, the block recursive motion estimation algorithm is derived as following. Assume that changes of the luminance signal in the image plane from frame k to frame $k+1$ are only due to the zoom and pan of the camera and the translatory motion of moving object. Let the luminance at position (x,y) in frame k be defined as $S_k(x,y)$. If the motion model is assumed to be $A(x, y) = (a_1x + a_2, a_1y + a_3)$, then the relationship between $S_k(x,y)$ and $S_{k+1}(x, y)$ can be described as

$$S_{k+1}(x,y) = S_k(A(x,y)) \qquad (6)$$

Where a_1, a_2 and a_3 are the motion parameters of the moving object. The objective of motion parameter estimation is to estimate these parameters based on the contents of $S_{k+1}(x,y)$ and $S_k(x,y)$. A common approach is to reduce the displaced frame difference of the two successive frames. The displaced frame difference(DFD) is defined as

$$\begin{aligned} DFD(x,y) &= S_{k+1}(x,y) - S_k(\widehat{A}(x,y))\\ &= S_{k+1}(x,y) - S_k(\hat{a}_1x+\hat{a}_2,\ \hat{a}_1y+\hat{a}_3) \end{aligned} \qquad (7)$$

where $\hat{a}_1, \hat{a}_2, \hat{a}_3$ are the current estimates of the true motion parameters.

If $\hat{a}_1, \hat{a}_2$ and $\hat{a}_3$ approach a_1, a_2 and a_3 respectively, then DFD will approach zero. The objective of the proposed algorithm is to use a block of pixels to derive the $\hat{a}_1, \hat{a}_2$ and $\hat{a}_3$ such that DFD is minimized. Since $S_{k+1}(x,y) = S_k(A(x,y))$, the DFD can be expressed as $S_k(A(x,y)) - S_k(\widehat{A}(x,y))$. Using the Taylor Series expansion to express the luminance function at point $p_O = A(x, y)$ and motion level $e= (\hat{a}_1, \hat{a}_2, \hat{a}_3)$ with respect to the motion parameter a_i, we obtain

$$S_k(A(x, y)) = S_k(\widehat{A}(x, y)) + \sum_{i=1}^{3}\frac{\partial S_k}{\partial a_i}\Big|_{po,e}(a_i - \hat{a}_i) + r(x, y) \qquad (8)$$

where $r(x,y)$ denote the higher order terms of the Taylor Series expasion.

Neglecting the higher order expansion $r(x,y)$, we have :

$$\begin{aligned} DFD(x,y) &= G_{a1}(x,y)(a_1-\hat{a}_1)+G_{a2}(x,y)(a_2-\hat{a}_2)\\ &\quad + G_{a3}(x,y)(a_3-\hat{a}_3) \end{aligned}$$

where $G_{a_i}(x,y) = \left.\frac{\partial S_k}{\partial a_i}\right|_{po,e}$ (9)

In order to describe Eqn.(9) directly from the image signal, the gradients with respect to the motion parameter can be replaced by the gradient in X and Y coordinates in the kth frame. By defining $(x', y') = A(x,y)$, we have

$$\left.\frac{\partial S_k}{\partial a_i}\right|_{po,e} = \frac{\partial S_k}{\partial x'}\frac{\partial x'}{\partial a_i}\Big|_{po,e} + \frac{\partial S_k}{\partial y'}\frac{\partial y'}{\partial a_i}\Big|_{po,e} \quad (10)$$

$$\frac{\partial x'}{\partial a_1} = x \qquad \frac{\partial x'}{\partial a_2} = 1 \qquad \frac{\partial x'}{\partial a_3} = 0$$
$$\frac{\partial y'}{\partial a_1} = y \qquad \frac{\partial y'}{\partial a_2} = 0 \qquad \frac{\partial y'}{\partial a_3} = 1 \quad (11)$$

Thus,

$$\begin{aligned} DFD(x,y) &= S_{k+1}(x,y) - S_k(\widehat{A}(x,y)) \\ &= S_k(A(x,y)) - S_k(\widehat{A}(x,y)) \\ &= (G_x x + G_y y)\times(a_1 - \hat{a}_1) + G_x \times (a_2 - \hat{a}_2) \\ &\quad + G_y \times (a_3 - \hat{a}_3) \\ &= (G_x x + G_y y)\, u_1 + G_x u_2 + G_y u_3 \end{aligned} \quad (12)$$

where $G_x = \left.\frac{\partial S_k}{\partial x'}\right|_{po,e}$ and $G_y = \left.\frac{\partial S_k}{\partial y'}\right|_{po,e}$ are X and Y directional gradient functions on point $\widehat{A}(x,y)$ and u_i is the difference between a_i and $\hat{a}_i$.

According to the steepest descent algorithm [8], a linear estimator to minimize the square value of DFD can be written as

$$\widehat{\mathbf{A}}^{(p+1)} = \widehat{\mathbf{A}}^{(p)} - \varepsilon' \mathbf{G}^{(p)} DFD^{(p)} \quad (13)$$

where

$$\widehat{\mathbf{A}}^{(p)} = [\hat{a}_1^{(p)} \;\; \hat{a}_2^{(p)} \;\; \hat{a}_3^{(p)}]^T$$

$$\mathbf{G}^{(p)} = [G_x x + G_y y \qquad G_x \qquad G_y]^T$$

with

$\widehat{\mathbf{A}}^{(p)}$ is the estimated motion parameter vector at iteration p.

ε' is the convergence factor of constant value.

Combining with the concept of block processing, a block recursive motion estimation algorithm is proposed by using N points of data to obtain a system of N linear equations to minimize the mean square value of DFD. This algorithm is based on the assumption that all pixels in a block have the same motion parameters. Writting in the matrix form we have

$$\widehat{\mathbf{A}}^{(p+1)} = \widehat{\mathbf{A}}^{(p)} - \varepsilon' \mathbf{R}^T \mathbf{D} \quad (14)$$

where

$$\mathbf{D} = \begin{pmatrix} DFD(x_1^{(p)}, y_1^{(p)}) \\ DFD(x_2^{(p)}, y_2^{(p)}) \\ \vdots \\ DFD(x_N^{(p)}, y_N^{(p)}) \end{pmatrix} \quad (15)$$

$$\mathbf{R} = \begin{pmatrix} G_{x_1(p)} x_1 + G_{y_1(p)} y_1 & G_{x_1(p)} & G_{y_1(p)} \\ G_{x_2(p)} x_2 + G_{y_2(p)} y_2 & G_{x_2(p)} & G_{y_2(p)} \\ \vdots & \vdots & \vdots \\ G_{x_N(p)} x_N + G_{y_N(p)} y_N & G_{x_N(p)} & G_{y_N(p)} \end{pmatrix} \quad (16)$$

and

p is the iteration index of a block.

$\mathbf{D}$ is the displaced frame difference vector.

$G_{xi}^{(p)} = G_x(x_i^{(p)}, y_i^{(p)})$ and $G_{yi}^{(p)} = G_y(x_i^{(p)}, y_i^{(p)})$ are the X and Y directional gradient values at the corresponding point $(x_i^{(p)}, y_i^{(p)})$ in frame k+1.

2.3 the modification of block recursive algorithm

If we apply the updating function of motion parameters derived in Eqn.(14) into Eqn.(1) to find the corresponding coordinate of pixels, we get

$$\begin{aligned} x_i^{(p+1)} &= \hat{a}_1^{(p+1)} x_i + \hat{a}_2^{(p+1)} \\ &= \{\hat{a}_1^{(p)} + \varepsilon' \sum_{i=1}^{N} [(G_{x_i}^{(p)} x_i + G_{y_i}^{(p)} y_i) DFD_i]\} x_i + [\hat{a}_2^{(p)} + \varepsilon' \sum_{i=1}^{N} (G_{x_i}^{(p)} DFD_i)] \\ &= \hat{a}_1^{(p)} x_i + \hat{a}_2^{(p)} + \varepsilon' \sum_{i=1}^{N} \{[(G_{x_i}^{(p)} \bullet x_i \bullet x_i + G_{y_i}^{(p)} \bullet y_i \bullet x_i) + G_{x_i}^{(p)}] DFD_i\} \\ &= x_i^{(p)} + \varepsilon' \sum_{i=1}^{N} [(G_{x_i}^{(p)} \bullet x_i^2 + G_{y_i}^{(p)} \bullet x_i \bullet y_i + G_{x_i}^{(p)}) DFD_i] \end{aligned}$$

and

$$\begin{aligned} y_i^{(p+1)} &= \hat{a}_1^{(p+1)} y_i + \hat{a}_3^{(p+1)} \\ &= \{\hat{a}_1^{(p)} + \varepsilon' \sum_{i=1}^{N} [(G_{x_i}^{(p)} x_i + G_{y_i}^{(p)} y_i) DFD_i]\} y_i + [\hat{a}_3^{(p)} + \varepsilon' \sum_{i=1}^{N} (G_{y_i}^{(p)} DFD_i)] \\ &= \hat{a}_1^{(p)} y_i + \hat{a}_3^{(p)} + \varepsilon' \sum_{i=1}^{N} \{[(G_{x_i}^{(p)} \bullet x_i \bullet y_i + G_{y_i}^{(p)} \bullet y_i \bullet y_i) + G_{y_i}^{(p)}] DFD_i\} \\ &= y_i^{(p)} + \varepsilon' \sum_{i=1}^{N} [(G_{x_i}^{(p)} \bullet x_i \bullet y_i + G_{y_i}^{(p)} \bullet y_i^2 + G_{y_i}^{(p)}) DFD_i] \end{aligned} \quad (17)$$

From Eqn.(17) we could find the update term of motion vector is propotional to the coordinates of pixel. The position of block is an important factor to influence the stability of the estimation algorithm. The estimated motion parameters of block with bigger coordinates may get a large change and

then diverge. If we decrease the convergence factor to increase the stability of algorithm, the convergence speed will be slowed down and so the recursive iteration must be increaed. With this, the convergence factor ε' in Eqn.(14) could be modified to be

$$\varepsilon' = \begin{pmatrix} \frac{\varepsilon_1}{|X_0 Y_0|} & 0 & 0 \\ 0 & \varepsilon_2 & 0 \\ 0 & 0 & \varepsilon_2 \end{pmatrix} \quad (18)$$

where ε_1 and ε_2 are the convergence factors for zooming and paaning motion parameters and (X_0, Y_0) is the coordinates of the center point of block.

With the modification on convergence factor the block recursive algorithm will efficiently increase the stability. But there is a special case to cause the modified algorithm to be unstable. This situation happens in that the the x-directional coordinate and y-directional coordinate are one a small value and another a big value. In this case the unbalanced coordinate will cause x/y or y/x to be a large number and so the update term of motion vector will be large and cause the algorithm to be unstable. To improve this, the convergence factor matrix will be changed as

$$\varepsilon' = \begin{pmatrix} \frac{\varepsilon_1}{|X_0+c|\,|Y_0+c|} & 0 & 0 \\ 0 & \varepsilon_2 & 0 \\ 0 & 0 & \varepsilon_2 \end{pmatrix} \quad (19)$$

where c is a constant value.

In Eqn.(19) the constant c is used to make the value $x/(y+c)$ or $y/(x+c)$ not to be so large to cause a unstable result. Another method for the improvement of the convergence factor is

$$\varepsilon' = \begin{pmatrix} \frac{\varepsilon_1}{X_0^2+Y_0^2} & 0 & 0 \\ 0 & \varepsilon_2 & 0 \\ 0 & 0 & \varepsilon_2 \end{pmatrix} \quad (20)$$

To see the performance of the block recursive algorithm two set of experiments are made for comparison. The image named as "Laboratory" is used for simulation and shown in Fig.1. The desired motion vector is (1.05, 2, 1). In these experiments the block with size of 8×8 is used to find the motion parameters. In the first set of experiments the output performances of the pel-recursive process and the block recursive process after 50 estimation iterations are compared. The blocks with center at (5, 5) and (-5, -5) are used for simulation. The simulation results are in Table.1. In Table.1 we can see the block process method can get the more accurate results than the single pixel process method. In the second set of experiments the output performances of the block recursive algorithm with different modifications are compared. There are two cases for simulation. In the first case blocks with large x-directional coordinate and large y-directional coordinate are used. The centers of block are at (51, 51) and (59, 59). The simulation results are shown in Table.2 and Fig.2. In Table.2 it shows the estimated motion parameters and DFD of different methods after 50 estimation iterations. In Fig.2 it shows the changing curve of the estimated zooming parameter to see the stability of algorithm. The modification methods derived in Eqn.(18) to Eqn.(20) are labeled as modification1, modification2 and modification3 respectively. According to the simulation results we could find the estimated motion parameters of the block recursive algorithm with no modification will get a divergent result. After using the modification on convergence factor, the stability of algorithm is clearly improved. The modification2 method gets the better stability than the modification1 method. The modification3 method gets the best stability but the slowest convergence speed. In the second case the blocks with large x-directional coordinate and small y-directional coordinate are used for simulation. The center positions of block are at (59, 3) and (83, 3). These results are shown in Table.3 and Fig.3. According to the results it could be found the modification1 method gets the unstable results. The modification3 method also gets the best stability but the slowest convergence speed. The usage of the modification2 method or the modification3 method is a trade between the stability and the convergence speed. In the pel-recursive algorithm proposed in [4] it has made a modification on convergence factor for the large value of gradient. But in our analysis, most of the gradients in real image sequence is kept to be a small value. So under the consideration of calculation amount the modification for the large value of gradient is not made in our proposed.

algorithm	position	a_1	a_2	a_3	DFD
block recursive	(5, 5)	1.052	2.3	0.9	6.56
	(-5, -5)	1.050	2.0	1.2	5.41
pel-recursive	(5, 5)	1.020	0.9	0.9	4.11
	(-5, -5)	1.022	1.1	0.6	4.10

Table.1 The simulation results of the block recursive algorithm and the pel recursive algorithm.

algorithm	a_1	a_2	a_3	DFD
original	0.0	20	20	3842.43
modification1	1.057	1.7	0.6	10.12
modification2	1.053	1.7	0.7	9.89
modification3	1.050	1.9	0.9	9.50

(a)

algorithm	a_1	a_2	a_3	DFD
original	5.0	-12.3	-20	6732.00
modification1	1.056	1.5	1.1	15.76
modification2	1.054	1.8	1.2	11.55
modification3	1.052	2.1	0.9	11.31

(b)

Table.2 The simulation results for blocks with center at (a) (51, 51) (b) (59, 59) with different modifications.

algorithm	a_1	a_2	a_3	DFD
modification1	0.0	7.3	8.7	133.56
modification2	1.056	1.6	0.7	13.78
modification3	1.052	1.8	0.8	7.88

(a)

algorithm	a_1	a_2	a_3	DFD
modification1	0.0	15.5	19.0	945.76
modification2	1.065	2.5	1.6	25.31
modification3	1.055	2.2	1.1	15.20

(b)

Table.3 The simulation results for blocks with center at (a) (59, 3) (b) (83, 3) with different modifications.

After making a modification on convergence factor for the stability, the block recursive algorithm still meets the problem of large amount of calculation. The complex computation makes the block recursive algorithm hard to be used for the real time application. So some simplifications must be made to slove this problem.

III The block recursive algorithm with finite word length

3.1 The quantized block recursive algorithm

As the constraint of the technique of image capturing or photoing, finite word length is the constraint on the quality of image of the digital system. To increase the quality of image picture the bit number of every pixel must be increased. On the other way, in order to increase the speed of processing or to decrease the size of memory for storing the image datas, the bit number of pixel must be decreased. So how to choose the word length for digital signal process system to get the best output performance is a important topic in disigning the signal processing system. Recently the quantized state adaptive algorithm has been sparked by the use of adaptive filters in high speed data computations and speech processing where the high data rate requires a computationally simple algorithm with large dynamic range. Using the concept of quantization, we simplify the block recursive algorithm by reducing the word length of some calculation parameters. The simplification strategy is to use the quantization on DFD or on gradient. The influences are introduced as following.

1. Quantization on DFD In the block recursive algorithm derived in section II the DFD is a important judgement of the output performance. The quantization on DFD will decrease the convergence accuracy of the estimation algorithm. On the other way the quantization on DFD will decrease the complexity of calculation $DFD(x_i^{(p)}, y^{(p)})G_{xi}^{(p)}$ or $DFD(x_i^{(p)}, y_i^{(p)})G_{yi}^{(p)}$ and so the complexity of the block recursive algorithm will be reduced.

2. Quantization on gradient If we quantize the gradient to be less word length but keep the direction of gradient to be similar, the performance of the quantized algorithm will make the least influence. Under this situation the convergence speed will be influenced to be slower but the convergence accuracy will be kept similar. At the same time the word length reduction of gradient will reduce the complexity of algorithm and increase the calculation speed.

Base on above analyses it could be found the quantization on gradient will reduce the complexity of algorithm with the least performance loss. So the quantization on gradient is used to simplify the block recursive estimation algorithm and Eqn.(16) will be replaced by

$$\mathbf{Q}=\begin{pmatrix} Q_{x_1}(p)x_1+Q_{y_1}(p)y_1 & Q_{x_1}(p) & Q_{y_1}(p) \\ Q_{x_2}(p)x_2+Q_{y_2}(p)y_2 & Q_{x_2}(p) & Q_{y_2}(p) \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ Q_{x_N}(p)x_N+Q_{y_N}(p)y_N & Q_{x_N}(p) & Q_{y_N}(p) \end{pmatrix} \tag{21}$$

where $Q_{xi}^{(p)}$, $Q_{yi}^{(p)}$ are the quantized values of $G_{xi}^{(p)}$, $G_{yi}^{(p)}$.

According to the analysis of the feature of image gradient, we find most of the gradients of pixel are small values. There are over 92 percent of gradients under the value of 32. According to the nonlinear distribution of gradient, a nonlinear quantization method is used for the quantiztion of gradient. The nonlinear quantization method to be used is to keep the first bit 1 of signal unchanged and change all the follwing bit 1s to be bit 0s. Taking the signal 01011010 as an example, the quantized signal will be 01000000. After using the nonlinear quantization on a 8-bit integer signal, the probability distribution of the quantization error e_g will be

$$\begin{cases} e_g=0, & prob=\dfrac{17}{511} \\ e_g=(2N+1-2N)+m, \quad 0\le N\le 6,\ 0\le m\le 2N-1, & prob=\dfrac{2\times(7-N)}{511} \end{cases} \tag{22}$$

According to the probability distribution in Eqn.(22), if we assume all the gradients are under value 32 and are normal distributed, the expective value of the error signal e_g will be 4.8. If the nonlinear distribution of the gradient values is taking into account, the expective value of quantization error will be less. Under the consideration of implementation, because there is at most one bit 1 in the quantized signal Q_x or Q_y, the multiplication with Q_x or Q_y could be replaced by the shift bit operation. So the multiplication amount needed for the calculation $\mathbf{Q}^T\mathbf{D}$ can be largely reduced.

There are some experimants made to see the output performance of block recursive algorithm with different word length of gradient. The image "Laboratory" is also used for simulation. The modification3 method in Eqn.(20) is used to get the better stability. The block with center at (25, 25) is used for simulation. The change curves of zooming motion parameter and mean square value of DFD are used to see the convergence speed and convergence accuracy. The simulation results are shown in Fig.4. In Fig.4 it shows the simulation results of the block recursive algorithm with the 5-bit to 8-bit linear quantization methods and the 3-bit

nonlinear quantization method. According to these simulation results we find that the reduction of word length of gradient only causes the convergence speed to be slower but convergence accuracy is kept similar. It also shows the convergence performance of the 3-bit nonlinear quantization method is similar to that of the 7-bit linear quantization method. This proves that the nonlinear quantization method used for the simplification of the block recursive algorithm can get good efficiency with the least performance loss.

3.2. The multiplication reduction according to the coordinate relationship

After using shift bit operation to replace the multiplication with the quantized gradient, the rest of multiplications needed for the calculation $\mathbf{Q}^T\mathbf{D}$ are the multiplications with X and Y directional coordinates. If we consider the relationship between the coordinates in a block, these multiplications can also be largely reduced. It will be discussed in the following.

Define the shift bit operation of the quantized gradient on *DFD* as

$$D_{xi}^{(p)} = DFD(x_i^{(p)}, y_i^{(p)}) \times Q_{yi}^{(p)}$$
$$D_{yi}^{(p)} = DFD(x_i^{(p)}, y_i^{(p)}) \times Q_{yi}^{(p)} \quad (23)$$

The calculation $\mathbf{Q}^T\mathbf{D}$ could be rewritten as

$$\mathbf{v} = [\, v_1, v_2, v_3 \,]^T = \mathbf{Q}^T\mathbf{D}$$

$$v_1 = \sum_{i=1}^{N} DFD(x_i^{(p)}, y_i^{(p)}) \times (Q_{x_i}{}^{(p)} x_i + Q_{y_i}{}^{(p)} y_i)$$
$$= \sum_{i=1}^{N} D_{x_i}{}^{(p)} \times x_i + D_{y_i}{}^{(p)} \times y_i$$
$$v_2 = \sum_{i=1}^{N} DFD(x_i^{(p)}, y_i^{(p)}) \times Q_{x_i}{}^{(p)} = \sum_{i=1}^{N} D_{x_i}{}^{(p)}$$
$$v_3 = \sum_{i=1}^{N} DFD(x_i^{(p)}, y_i^{(p)}) \times Q_{y_i}{}^{(p)} = \sum_{i=1}^{N} D_{y_i}{}^{(p)} \quad (24)$$

To complete the calculation of $\mathbf{Q}^T\mathbf{D}$, it remains the multiplications of coordinates x_i and y_i with $D_{xi}^{(p)}$ and $D_{yi}^{(p)}$. If we consider the relationship between the coordinates of pixels in a block, these multiplications can be largely reduced. Taking a 4×4 block as an example, the coordinate relationships between pixels are shown in Fig.5. It can be seen that the X coordinates of the pixels in the same column and The Y coordinates of the pixels in the same row are the same. So there are only four different X indices and four different Y indices for the 16 coordinates of pixels. Denoting the left-upper corner coordinate as (x, y), the rest of the element can be denoted as $(x+h, y-i)$, $h, i = 0,1,2,3$. By defining

the ith column summation as

$$Z_{x_i} = D_{x(1+i)}{}^{(p)} + D_{x(5+i)}{}^{(p)} + D_{x(9+i)}{}^{(p)} + D_{x(13+i)}{}^{(p)}$$

the ith row summation as

$$Z_{y_i} = D_{y(1+4i)}{}^{(p)} + D_{y(2+4i)}{}^{(p)} + D_{x(3+4i)}{}^{(p)} + D_{y(4+4i)}{}^{(p)} \quad (25)$$

The vector $\mathbf{v}$ for a 4×4 block can be calculated as

$$v_1 = (\sum_{i=0}^{3} Z_{x_i})x + (\sum_{i=0}^{3} Z_{y_i})y + \{(Z_{x_1}+Z_{x_3}) + 2\times(Z_{x_2}+Z_{x_3})\}$$
$$- \{(Z_{y_1}+Z_{y_3}) + 2\times(Z_{y_2}+Z_{y_3})\}$$
$$v_2 = \sum_{i=1}^{16} D_{x_i}{}^{(p)} = \sum_{i=0}^{3} Z_{x_i}$$
$$v_2 = \sum_{i=1}^{16} D_{y_i}{}^{(p)} = \sum_{i=0}^{3} Z_{y_i} \quad (26)$$

According to Eqn.(26), for a 4×4 block only 2 multiplications and 39 additions are needed to complete the calculation of $\mathbf{Q}^T\mathbf{D}$.

After get the new motion parameters, the addresses of the new corresponding points with the new estimated motion parameter $\hat{a}_i^{(p+1)}$ must be calculated. To get the new address of $(a_1 x_i + a_2, a_1 y_i + a_3)$, the multiplications with x_i and y_i are also needed. So the multiplication reduction according the coordinate relationship can also be used. According to above derivation, the computation requirement comparison of the original block recursive algorithm and the simplified method are shown in Table.3. According to the results we find the multiplications needed for the simplified algorithm will be over 30 times less than those needed for the original method. Such a large amount of calculation reduction makes the simplified block recursive algorithm easier to be implemented for high speed motion estimation.

operation	number of multiplication	number of addition
$\mathbf{R}^T\mathbf{D}$	64	45
$\mathbf{Q}^T\mathbf{D}$ with coordinate relationship	2	39

Table 4 The computational amount comparison

IV. Conclusion

A block recursive algorithm to compute the three motion parameters for video coding is presented in this paper. With the modification of coordinate on the convergence factor the modified algorithm get the better stability and convergence accuracy. To solve the problem of computational complexity, the quantized gradient is is used and proven to get good result with a little performance loss. Combining with the computational complexity reduction according to the coordinate relationships, the amount of calculations can be largely reduced and make the computational speed easier to meet the real time calculation requirement.

Reference

[1] M.Hotter, "Differential Estimation of The Global Motion Parameters Zoom and Pan", Signal Processing, vol.16, no.3, March 1989, pp.249-265.

[2] Y. Nakaya, H. Harashima, "Motion Compensation Based on Spatial Transformations", IEEE Trans. on CAS for Video Technology, vol.4, no.3, Jun. 1994, pp-339-356.
[3] M. Bierling, "Displacement estimation by hierarchical blockmatching", SPIE Vol.1001, visual communications and Image Processing'88, pp.942-951, 1988.
[4] Walker, D. R. and K. R. Rao, "Improved Pel-Recurssive Motion Compensation", IEEE Transactions on Commucations, vol. COM-32, no. 10, December 1984, pp.1128-1134.
[5] G. A. Clark, S. K. Mitra, and S. R. Parker, "Block Implementation of Adaptive Digital Filter", IEEE Trans. Circuits syst. , vol. CAS-28, June 1981, pp.584-592.
[6] "Adaptive Equalization via Fast Quantized-State Methods", IEEE Trans. Commun., vol. COM-29, Oct. 1981, pp.1492-1501.
[7] A. Gersho, "Adaptive filtering with Binary reinforcement", IEEE Trans. Inform. Theory, vol. IT-30, March 1984.
[8] W. A. Sethares and Richard Johnsen, JR. "A Comparison of Two Quantized State Adaptive Algorithms", IEEE Trans. Acoust., Speech, signal Processing, vol.37, No.1, Jan. 1989, pp.138-143.

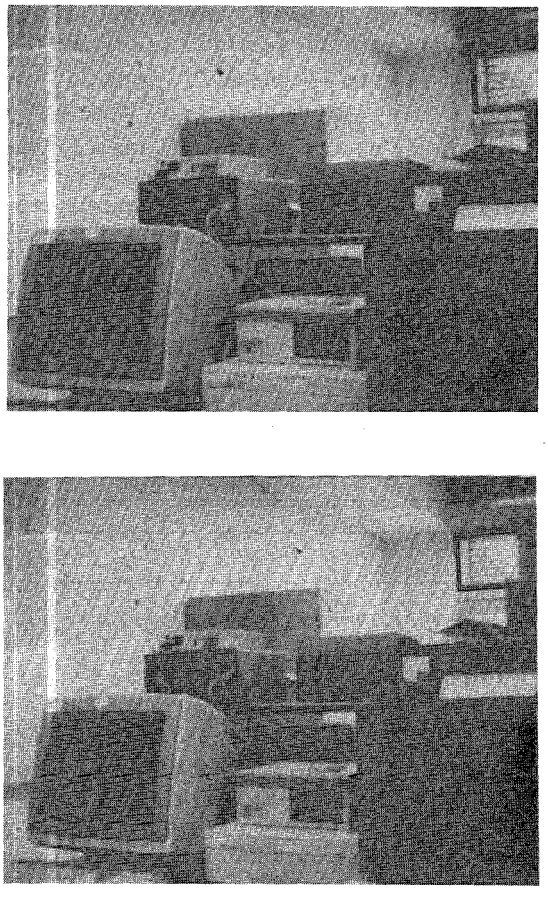
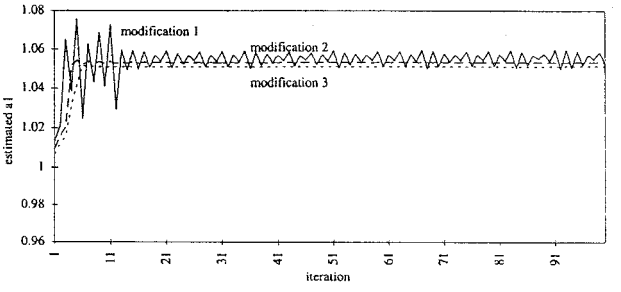
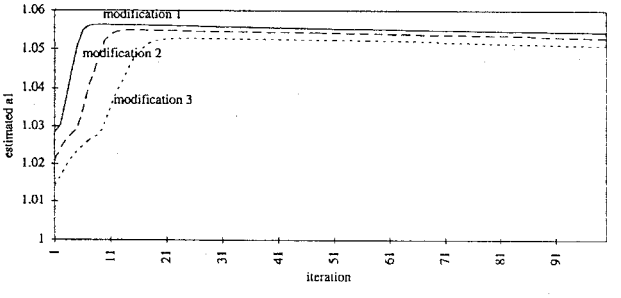


Fig.1 The simulated image "Laboratory".



(a)



(b)

Fig.2 The simulation result of zooming factor for blocks with center at (a) (51, 51) (b) (59, 59).

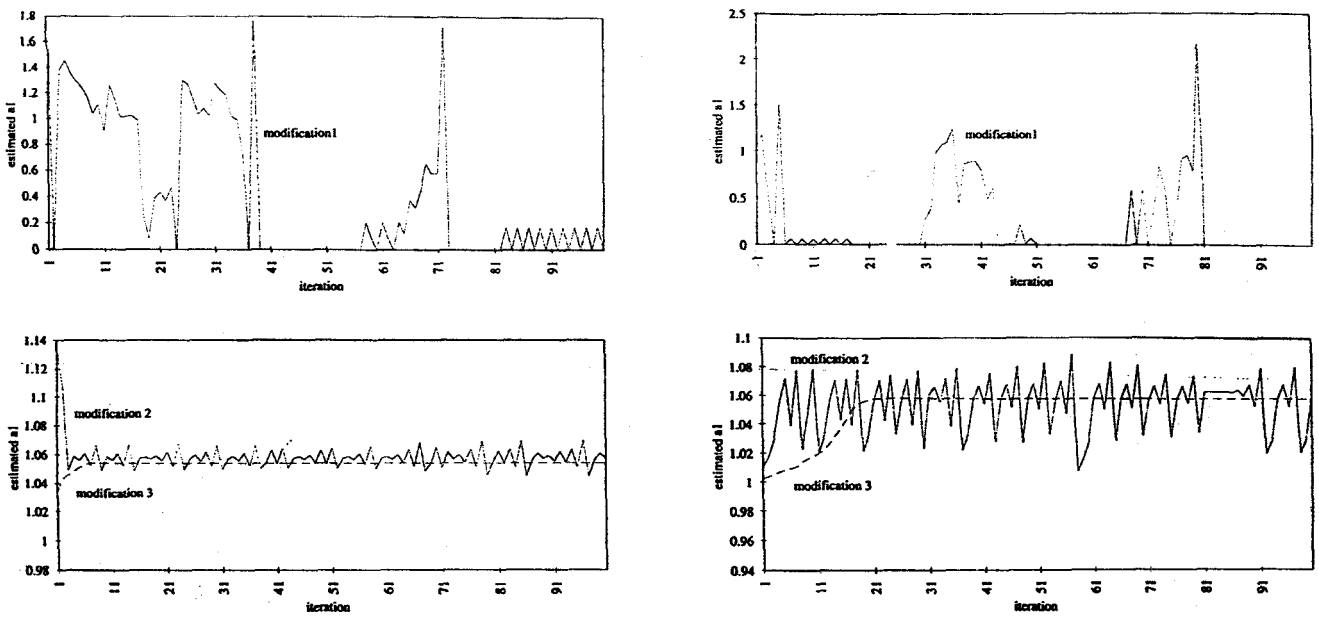


Fig.3 The simulation result of zooming factor for blocks with center at (a) (59, 3) (b) (83, 3).

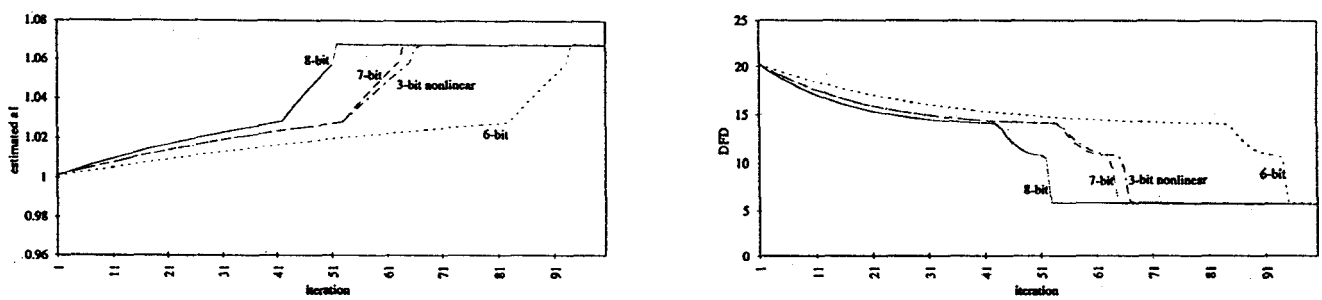


Fig.4 The simulation result of (a) zooming factor (b) DFD with linear and nonlinear quantization methods.

$(x1, y1)$ (x, y)	$(x2, y2)$ $(x+1, y)$	$(x3, y3)$ $(x+2, y)$	$(x4, y4)$ $(x+3, y)$
$(x5, y5)$ $(x, y-1)$	$(x6, y6)$ $(x+1, y-1)$	$(x7, y7)$ $(x+2, y-1)$	$(x8, y8)$ $(x+3, y-1)$
$(x9, y9)$ $(x, y-2)$	$(x10, y10)$ $(x+1, y-2)$	$(x11, y11)$ $(x+2, y-2)$	$(x12, y12)$ $(x+3, y-2)$
$(x13, y13)$ $(x, y-3)$	$(x14, y14)$ $(x+1, y-3)$	$(x15, y15)$ $(x+2, y-3)$	$(x16, y16)$ $(x+3, y-3)$

Fig.5 The realtionship of coordinates of a 4×4 rectangular block.