24

Kalman Filtering Based Motion Estimation for Video Coding

Assist. Prof. Nai-Chung Yang[1], Prof. Chaur Heh Hsieh[2]
and Prof. Chung Ming Kuo[1]
[1]*Dept. of information Engineering, I-Shou University, Dahsu Kaohsiung, 840,*
[2]*Dept. of Computer and Communication Engineering, Ming Chuan University,*
Gui-Shan, Taoyuan, 333,
Taiwan, R.O.C.

1. Introduction

Video compression is a very efficient method for storage and transmission of digital video signal. The applications include multimedia transmission, teleconferencing, videophone, high-definition television (HDTV), CD-ROM storages, etc. The hybrid coding techniques based on predictive and transform coding are the most popular and adopted by many video coding standards such as MPEG-1/2/4 [1] and H.261/H.263/H.264 [2, 3], owing to its high compression efficiency. In the hybrid coding system, the motion compensation, first proposed by Netravali and Robbins in 1997, plays a key role from the view point of coding efficiency and implementation cost [4-11]. A generic hybrid video coder is depicted in Figure 1.

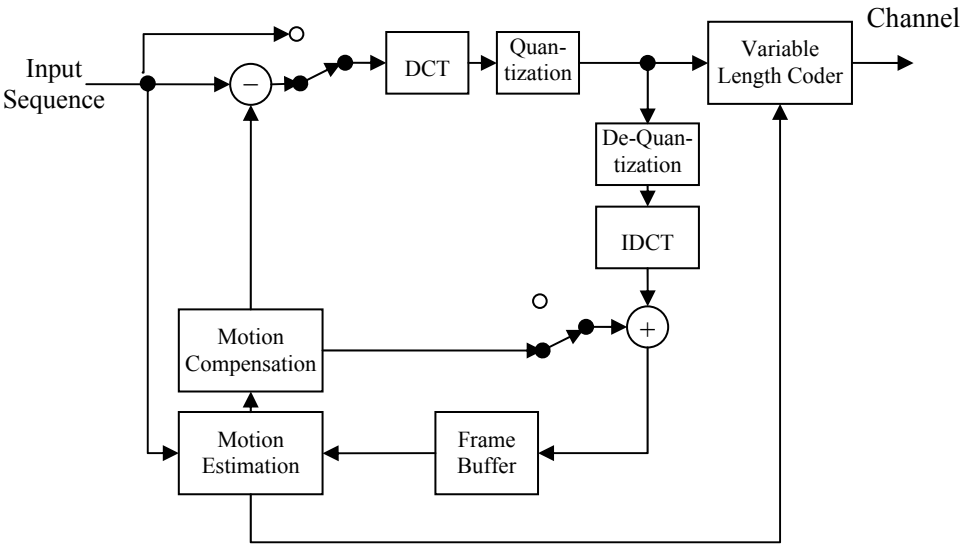


Fig. 1. A generic hybrid motion compensated DCT video coder.

Source: Kalman Filter: Recent Advances and Applications, Book edited by: Victor M. Moreno and Alberto Pigazo,
ISBN 978-953-307-000-1, pp. 584, April 2009, I-Tech, Vienna, Austria

The main idea of video compression to achieve compression is to remove spatial and temporal redundancies existing in video sequences. The temporal redundancy is usually removed by a motion compensated prediction scheme, whereas the spatial redundancy left in the prediction error is commonly reduced by a discrete cosine transform (DCT) coder. Motion compensated is a predictive technique in temporal direction, which compensates for the displacements of moving objects from the reference frame to the current frame. The displacement is obtained with the so-called motion vector estimation. Motion estimation obtains the motion compensated prediction by finding the motion vector (MV) between the reference frame and the current frame.

The most popular technique used for motion compensation (MC) is the block-matching algorithm (BMA) due to its simplicity and reasonable performance. In a typical BMA, the current frame of a video sequence is divided into non-overlapping square blocks of $N \times N$ pixels. For each reference block in the current frame, BMA searches for the best matched block within a search window of size $(2P+1)\times(2P+1)$ in the previous frame, where P stands for the maximum allowed displacement. Figure 2 depicts the basic principle of block matching.

In general, BMAs are affected by following factors: (i) search area, (ii) matching criterion, and (iii) searching scheme. The matching criterion is to measure the similarity between the block of the current frame and candidate block of the reference frame. Two typical matching criteria are mean square error (MSE) and mean absolute error (MAE), which are defined respectively as below:

$$MSE(u,v)=\frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\left[f(x,y,k)-f(x+u,y+v,k-1)\right]^2 \text{ , for } u,v\in[-P,P]$$

$$MAE(u,v)=\frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\left|f(x,y,k)-f(x+u,y+v,k-1)\right| \text{ , for } u,v\in[-P,P]$$

where $f(x,y,k)$ denotes the coordinate of the top left corner of the searching block of the current frame k, and (u,v) is the displacement of the matching block of frame $k-1$. The MAE is the most popular matching criterion due to its simplicity of hardware implementation.

The searching scheme is very important because it is significantly related to with the computational complexity and accuracy of motion estimation for general video applications. A straightforward way to obtain the motion vector is the full search algorithm (FSA), which searches all locations in the search window and selects the position with minimal matching error. However, its high computational complexity makes it often not suitable for real-time implementation. Therefore, many fast search algorithms have been developed to reduce the computational cost. In general, fast search algorithms reduce the computational burden by limiting the number of search locations or by sub-sampling the pixels of a block. However, they often converge to a local minimum, which leads to worse performance.

Most search algorithms estimate the motion vector (MV) for each block independently. In general moving scenes, it is very likely that a large homogeneous area in the picture frame will move in the same direction with similar velocities. Therefore, the displacements between neighboring blocks are highly correlated. Some schemes take advantage of this correlation to reduce the computational complexity [14-16].

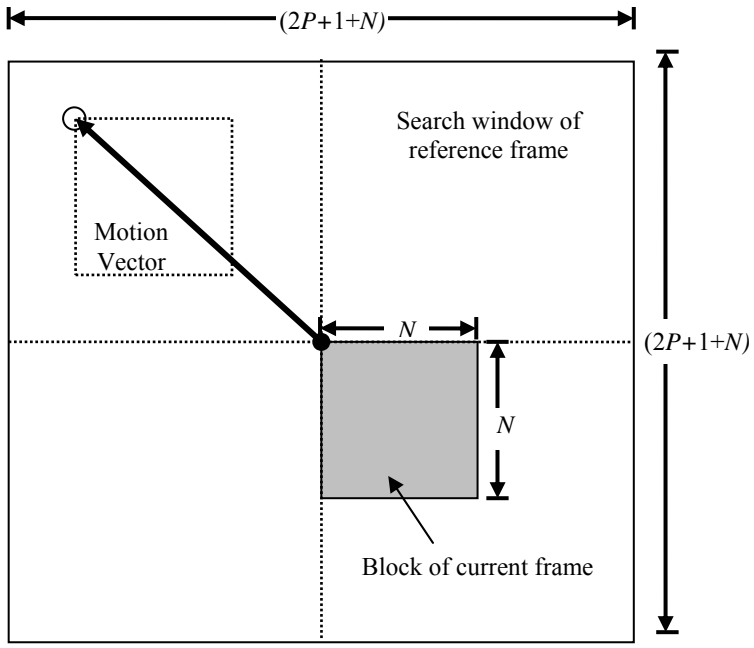


Fig. 2. Block matching algorithm.

There are two major problems for the existing fast search algorithms. One is that the estimation accuracy in terms of the energy or the entropy of the motion-compensated prediction error (MCPE) signal is worse than that of FSA. The other is that the true motion may not be obtained even with FSA, which is very important in some applications such as motion compensated interpolation and frame (field) rate conversion. Bierling [17] proposed a hierarchical search scheme to achieve a truer (smoother) motion vector field over FSA, but it results in a worse performance in terms of the energy of the MCPE signal.

The above two problems may arise from the following reasons [18]: (i) the basic assumptions, including pure translation, unchanged illumination in consecutive frames and noiseless environment, are not exactly correct; furthermore, another assumption that the occlusion of one object by another and uncovered background are neglected is also not exactly correct, (ii) the size of a moving object may not be equal to the prescribed block size, (iii) the fast search schemes often converge to a local optimum. In Section 2, we will introduce how to overcome these problems with a relatively low computational cost. We neither relax the above assumptions nor develop a globally optimal search scheme. Instead, we use the Kalman filter to compensate the incorrect and/or inaccurate estimates of motion. We first obtain a measurement of motion vector of a block by using a conventional fast search scheme. We then generate the predicted motion vector utilizing the motion correlation between spatial neighboring blocks. Based on the predicted and measured motion information, a Kalman filter is employed to obtain the optimal estimate of motion vector. In the new method, a local Kalman filter is developed, which is based on a novel

motion model that exploits both spatial and temporal motion correlations. The proposed local Kalman filter successfully addresses the difficulty of multi-dimensional state space representation, and thus it is simpler and more computationally efficient than the conventional 2-D Kalman filter such as reduced update Kalman filter (RUKF) [19]. In addition, we will also introduce an adaptive scheme to further improve estimate accuracy while without sending extra side information to the decoder.

In low- or very low- bit rate applications such as videoconference and videophone, the percentage of MV bit rate increases when overall rate budget decreases. Thus, the coding of MVs takes up a significant portion of the bandwidth [20]. Then in very low bit rate compression, the motion compensation must consider the assigned MV rate simultaneously. A joint rate and distortion (R-D) optimal motion estimation has been developed to achieve the trade-off between MV coding and residue coding [20-28]. In [25], a global optimum R-D motion estimation scheme is developed. The scheme achieves significant improvement of performance, but it employs Viterbi algorithm for optimization, which is very complicated and results in a significant time delay. In [26], a local optimum R-D motion estimation criterion was presented. It effectively reduces the complexity at the cost of performance degradation.

In Section 3, we will introduce two Kalman filter-based methods to improve the conventional R-D motion estimation, which are referred to as enhanced algorithm and embedded algorithm, respectively. In the enhanced algorithm, the Kalman filter is employed as a post processing of MV, which extends the integer-pixel accuracy of MV to fractional-pixel accuracy, thus enhancing the performance of motion compensation. Because the Kalman filter exists in both encoder and decoder, the method achieves higher compensation quality without increasing the bit rate for MV.

In the embedded algorithm, the Kalman filter is applied directly during the process of optimization of motion estimation. Since the R-D motion estimation consider compensation error (distortion) and bit rate simultaneously, when Kalman filter is applied the distortion will be reduced, and thus lowering the cost function. Therefore, the embedded algorithm can improve distortion and bit rate simultaneously. Specifically, this approach can be combined with existing advanced motion estimation algorithms such as overlapped block motion compensation (OBMC) [29,30], and those recommended in H.264 or MPEG-4 AVC [31, 32].

2. Motion estimation with Kalman filter

2.1 Review of Kalman filter

The Kalman filtering algorithm estimates the states of a system from noisy measurement [33-36]. There are two major features in Kalman filter. One is its mathematical formulation is described in terms of state-space representation, and the other is that its solution is computed recursively. It consists of two consecutive stages: prediction and updating. We summarize the Kaman filter algorithm as follows:

$$\text{Predicted equation: } \mathbf{v}(k) = \mathbf{\Phi}(k-1)\mathbf{v}(k-1) + \mathbf{\Gamma}(k)\mathbf{w}(k), \tag{1}$$

$$\text{Measurement equation: } \mathbf{z}(k) = \mathbf{H}(k)\mathbf{v}(k) + \mathbf{n}(k), \tag{2}$$

where $\mathbf{v}(k)$ and $\mathbf{z}(k)$ are state and measurement vector at time k, and □,H and □ are state transition, measurement and driving matrix, respectively. The model error $\mathbf{w}(k)$, with covariance matrix $\mathbf{Q}(k)$, and measurement error $\mathbf{n}(k)$, with covariance matrix $\mathbf{R}(k)$, are often assumed to be Gaussian white noises; we assume that $\mathbf{w}(k)$~N(0,$\mathbf{Q}(k)$), $\mathbf{n}(k)$~N(0,$\mathbf{R}(k)$) and $E[\mathbf{w}(k)\mathbf{n}^T(l)]=0$ for all k and l. Let $E[\mathbf{v}(0)] = \hat{\mathbf{v}}(0)$, and $E[(\mathbf{v}(0) - \hat{\mathbf{v}}(0))(\mathbf{v}(0) - \hat{\mathbf{v}}(0))^T] = \mathbf{P}(0)$ be initial values. The prediction and updating are given as follows.

Prediction:

$$\text{State prediction: } \hat{\mathbf{v}}^-(k) = \mathbf{\Phi}(k-1)\hat{\mathbf{v}}^+(k-1) \tag{3}$$

Prediction-error covariance:

$$\mathbf{P}^-(k) = \mathbf{\Phi}(k-1)\mathbf{P}^+(k-1)\mathbf{\Phi}^T(k-1) + \mathbf{\Gamma}(k)\mathbf{Q}(k-1)\mathbf{\Gamma}^T(k) \tag{4}$$

Updating:

$$\text{State updating: } \hat{\mathbf{v}}^+(k) = \hat{\mathbf{v}}^-(k) + \mathbf{K}(k)[\mathbf{z}(k) - \mathbf{H}(k)\hat{\mathbf{v}}^-(k)] \tag{5}$$

$$\text{Updating-error covariance: } \mathbf{P}^+(k) = [\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)]\mathbf{P}^-(k) \tag{6}$$

$$\text{Kalman gain matrix: } \mathbf{K}(k) = \mathbf{P}^-(k)\mathbf{H}(k)[\mathbf{H}(k)\mathbf{P}^-(k)\mathbf{H}^T(k) + \mathbf{R}(k)]^{-1} \tag{7}$$

The $\mathbf{P}(k)$ is the error covariance matrix that is associated with the state estimate $\mathbf{v}$(k), and is defined as

$$\mathbf{P}(k) = E[(\mathbf{v}(k) - \hat{\mathbf{v}}(k))(\mathbf{v}(k) - \hat{\mathbf{v}}(k))^T]. \tag{8}$$

The superscripts "-" and "+" denote "before" and "after" measurement, respectively. The error covariance matrix $\mathbf{P}(k)$ provides a statistical measure of the uncertainty in $\mathbf{v}(k)$.

2.2 The overview of motion estimation with Kalman filter

In general, for moving scenes, the motion vectors among neighboring blocks are highly correlated. Therefore, the MV of the current block can be predicted from its neighboring blocks if an appropriate motion model is employed. Furthermore, any existing searching algorithms can be used to measure the MV. Using the predicted MV and the measured MV, a motion estimation method was developed, as depicted in Figure 3. The MV obtained with any conventional searching algorithm is defined as measurement, $\mathbf{z}(k)$. The measurement is then inputted to the Kalman filter and the updating estimate of MV could be obtained [37]. Because an identical Kalman filter will be used in the decoder, we can only send z(k), which is an integer, instead of $\hat{\mathbf{v}}(k)$, which is a real in general, to the receiver. By the same procedure, we can estimate $\hat{\mathbf{v}}(k)$ in the receiver, therefore we can achieve fractional-pixel accuracy with the bit rate of integer motion vector. In summary, there are two advantages for the new method: (i) it improves the performance of any conventional motion estimation due to the fractional-pixel accuracy; (ii) the transmitted bit rate for the motion vector is the

same as that of the input integer motion vector, therefore, the new method is compatible with the current video coding standards.

In the following, we will first introduce a motion model that exploits both spatial and temporal motion correlations, and then a local Kalman filter is developed accordingly. The local Kalman filter is simpler and more computationally efficient than the conventional 2-D Kalman filter such as RUKF. Therefore, it is more suitable for the real-time applications [48-49]. In addition, to further improve the motion estimate accuracy, we also introduce an adaptive scheme. The scheme can automatically adjust the uncertainty of prediction and measurement; however, it needs not to send side information to the decoder.

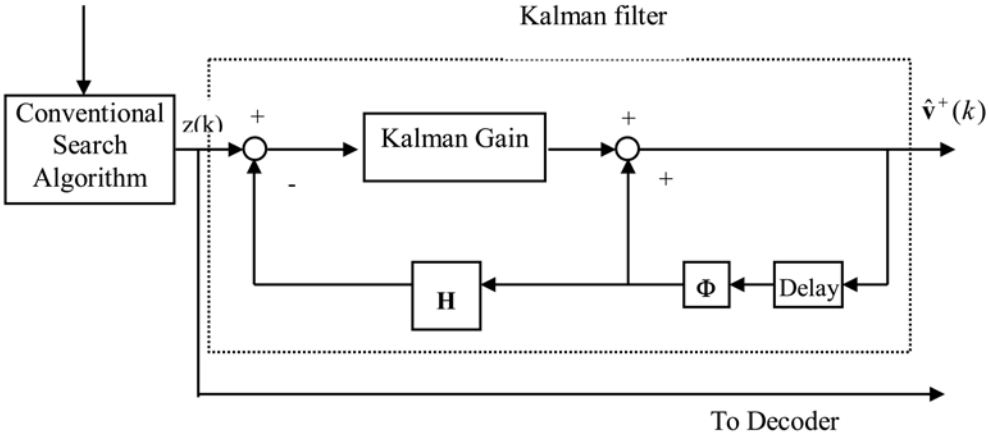


Fig. 3. Block diagram of motion estimation with Kalman filter

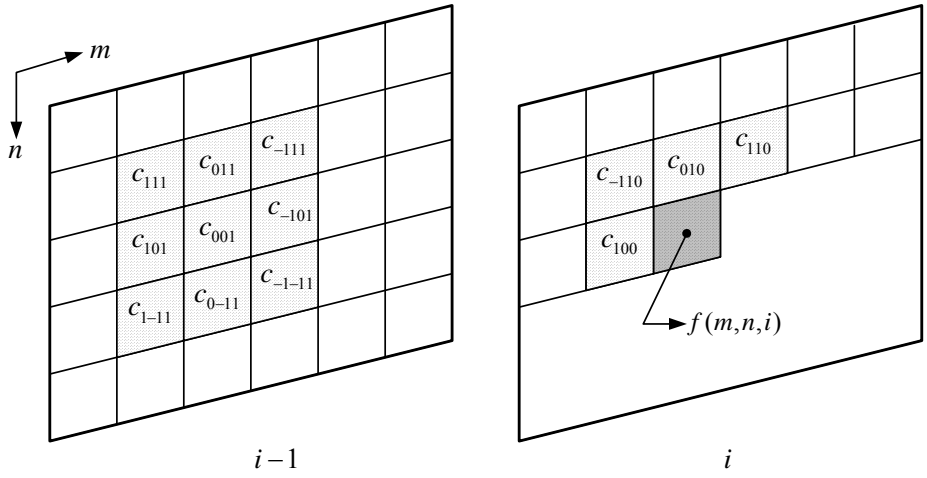
2.3 Motion estimation using Local Kalman Filter (LKF)

Let B(*m*,*n*,*i*) be the block at the location (*m*,*n*) in the *i*th frame, and $\mathbf{V}(m,n,i)=[v_x(m,n,i),v_y(m,n,i)]^T$ be the MV of B(*m*,*n*,*i*), where $v_x(m,n,i)$ and $v_y(m,n,i)$ denote the horizontal and vertical components, respectively. Assume that the MV is a random process, and the two components are independent. Then we can model these two components separately. In this work, we present a three-dimensional (3-D) AR model that exploits the relationship of motion vectors for only 3-D neighboring blocks that arrive at before the current block. We only choose the nearest neighboring blocks, in which the motion vectors are strongly correlated. We refer to this model as 3-D local model, which is expressed as

$$v_x(m,n,i)=\sum\sum\sum_{(k,l,p)\in S^{\oplus}} a_{klp}v_x(m-k,n-l,i-p)+w_x(m,n,i), \tag{9}$$

$$v_y(m,n,i)=\sum\sum\sum_{(k,l,p)\in S^{\oplus}} a_{klp}v_y(m-k,n-l,i-p)+w_y(m,n,i), \tag{10}$$

where $S^{\oplus}=\{l=0,k=1,p=0\}\cup\{l=1,|k|\le 1,p=0\}\cup\{|l|\le 1,|k|\le 1,p=1\}$. The support of the model mentioned above is depicted in Figure 4.



a) b)

Fig. 4. Causal AR models for motion vector associated with spatial and temporal neighborhood system.

2.3.1 State space representation of MV model

For the fully state propagation, we must represent the proposed models of Eqs. (9) and (10) in a state space. This will yield a 13-dimensional state vector. The high-dimension state vector will result in a huge computation for estimating the motion vector. To attack the computation problem, we decompose the AR model into two parts: filtering and prediction. The prediction part will not affect the state propagation; thus it is considered as a deterministic input. Consequently, the state space representation can be formulated as

$$\mathbf{v}(m,n,i) = \Phi\mathbf{v}(m-1,n,i) + \Lambda\mathbf{u}(m,n,i) + \Gamma\mathbf{w}(m,n,i)\,, \tag{11}$$

$$\mathbf{z}(m,n,i) = \mathrm{H}\mathbf{v}(m,n,i) + \mathbf{e}(m,n,i)\,. \tag{12}$$

In the above equations, $\mathbf{v}(m,n,i)$ represents the state vector at the location (m,n,i); $\mathrm{u}(m,n,i)$ denotes a deterministic input; and $\mathbf{\Phi}, \mathbf{\Lambda}, \mathbf{\Gamma}$ and $\mathbf{H}$ are the corresponding matrices. In our work, the deterministic input is defined as the prediction part of the model, which will be used to implement the local Kalman filter (LKF). Since the motion estimation processes the block one by one according to the order of raster scan, the state propagation should be performed in one-dimensional manner, as depicted in Eq. (11).

The main idea in LKF is the approximation of the MV $\mathbf{v}(m,n,i)$, which can not be represented in terms of $\mathbf{v}(m\text{-}1,n,i)$. We will demonstrate the state space representation in Eqs. (13) and (14) as follows.

$$\mathbf{v}(m,n,i) = \Phi\mathbf{v}(m-1,n,i) + \Lambda\mathbf{u}(m,n,i) + \Gamma\mathbf{w}(m,n,i) \tag{13}$$

where

$$\mathbf{v}(m,n,i)=\begin{bmatrix} v(m,n,i) \\ v(m-1,n,i) \\ v(m+2,n-1,i) \\ v(m+1,n-1,i) \\ v(m,n-1,i) \\ v(m+1,n,i-1) \end{bmatrix}, \mathbf{v}(m-1,n,i)=\begin{bmatrix} v(m-1,n,i) \\ v(m-2,n,i) \\ v(m-1,n-1,i) \\ v(m,n-1,i) \\ v(m-1,n,i) \\ v(m,n,i-1) \end{bmatrix}, \mathbf{u}(m,n,i)=\begin{bmatrix} v(m+1,n-1,i-1) \\ v(m,n-1,i-1) \\ v(m-1,n-1,i-1) \\ v(m+1,n,i-1) \\ v(m-1,n,i-1) \\ v(m+1,n+1,i-1) \\ v(m,n+1,i-1) \\ v(m-1,n+1,i-1) \end{bmatrix},$$

$$\mathbf{w}(m,n,i)=\begin{bmatrix} w(m,n,i) \\ w(m+1,n,i-1) \end{bmatrix},$$

$$\mathbf{\Phi}=\begin{bmatrix} c_{100} & 0 & c_{-110} & c_{010} & c_{110} & c_{001} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{\Lambda}=\begin{bmatrix} c_{-111} & c_{011} & c_{111} & c_{-101} & c_{101} & c_{-1-11} & c_{0-11} & c_{1-11} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and $\mathbf{\Gamma}=\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$.

$$\mathbf{z}(m,n,i)=\mathbf{H}\mathbf{v}(m,n,i)+\mathbf{e}(m,n,i) \tag{14}$$

where $\mathbf{z}(m,n,i)=\begin{bmatrix} z(m,n,i) \\ z(m+2,n-1,i) \end{bmatrix}$, $\mathbf{H}=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$ and $\mathbf{e}(m,n,i)=\begin{bmatrix} e(m,n,i) \\ e(m+2,n-1,i) \end{bmatrix}$.

The motion vector $\mathbf{v}(m,n,i)$, which can not be represented in terms of $\mathbf{v}(m\text{-}1,n,i)$, consists of two components: $v(m\text{+}2,n\text{-}1,i)$ and $v(m\text{+}1,n,i\text{-}1)$. We use the most recent estimate with uncertainty to approximate them, i.e.,

$$v(m+2,n-1,i)=\hat{v}(m+2,n-1,i)+e(m+2,n-1,i), \tag{15}$$

$$v(m+1,n,i-1)=\hat{v}(m+1,n,i-1)+w(m+1,n,i-1), \tag{16}$$

The above equations indicate that the best available estimate is the most recent update of the MV, which is available at time (m,n,i). The current frame MV, $v(m\text{+}2,n\text{-}1,i)$, is incorporated into measurement, and the previous frame MV, $v(m\text{+}1,n,i\text{ -}1)$, is incorporated into deterministic input. In our work, the covariance of these two uncertainties is given a small

value for simplicity. Through the above process, the motion estimation with 3-D AR model can be realized by 1-D recursive manner.
Given these models, the Kalman filter is described in the following:
<1> Prediction

$$\text{State prediction: } \hat{\mathbf{v}}^{-}(m,n,i) = \Phi\hat{\mathbf{v}}^{+}(m-1,n,i) + \Lambda\mathbf{u}(m,n,i) \tag{17}$$

$$\text{Prediction-error covariance: } \mathbf{P}^{-}(m,n,i) = \Phi\mathbf{P}^{+}(m-1,n,i)\Phi^{T} + \Gamma\,\mathbf{Q}(m,n,i)\Gamma^{T} \tag{18}$$

<2> Updating:

$$\text{State updating: } \hat{\mathbf{v}}^{+}(m,n,i) = \hat{\mathbf{v}}^{-}(m,n,i) + \mathbf{K}(m,n,i)[\mathbf{z}(m,n,i) - \mathbf{H}\hat{\mathbf{v}}^{-}(m,n,i)] \tag{19}$$

$$\text{Updating-error covariance: } \mathbf{P}^{+}(m,n,i) = [\mathbf{I} - \mathbf{K}(m,n,i)\mathbf{H}]\mathbf{P}^{-}(m,n,i) \tag{20}$$

$$\text{Kalman gain matrix: } \mathbf{K}(m,n,i) = \mathbf{P}^{-}(m,n,i)\mathbf{H}^{T}[\mathbf{H}\mathbf{P}^{-}(m,n,i)\mathbf{H}^{T} + \mathbf{R}(m,n,i)]^{-1} \tag{21}$$

The $\mathbf{P}(m,n,i)$ is the error covariance matrix that is associated with the state estimate $\mathbf{v}(m,n,i)$, $\mathbf{R}(m,n,i)$ and $\mathbf{Q}(m,n,i)$ are the covariance of $\mathbf{e}(m,n,i)$ and $\mathbf{w}(m,n,i)$, respectively.
However, the local model can be simplified to consider only spatial or temporal support, and then the motion model and the corresponding state space representation are modified accordingly.

2.3.2 Spatial causal AR models for MV

Let $B(m,n,i)$ be the block at the location (m,n) in the ith frame, and $\mathbf{V}(m,n,i)=[v_x(m,n,i),v_y(m,n,i)]^T$ be the MV of $B(m,n,i)$, where $v_x(m,n,i)$ and $v_y(m,n,i)$ denote the horizontal and vertical components, respectively. Assume that the MV is a random process, and the two components are independent. A 2-D AR model exploits the motion information of only 2-D neighboring blocks that arrived before the current block. In the block matching, the calculation of matching criterion is performed block-by-block in a raster scan manner, i.e., from left to right and top to bottom. Thus we can define the 2-D AR model for a motion vector as

$$v_{i,x} = \sum \sum_{(k,l)\in S^{+}} a_{kl0} v_{i,x}(m-k,n-l,i) + w_{i,x}(m,n,i)\,, \tag{22}$$

$$v_{i,y} = \sum \sum_{(k,l)\in S^{+}} a_{kl0} v_{i,y}(m-k,n-l,i) + w_{i,y}(m,n,i)\,, \tag{23}$$

where $S^{+} = \{k \geq l,\ \forall l\} \cup \{k = 0,\ l \geq 1\}$ is the model support, and a_{kl0} are the model coefficients, which can be space varying or space invariant. For simplicity, we assume that the model is space invariant. Eq. (22) and (23) are also called the nonsymmetric half-plane (NSHP) model [19].
We only chose the nearest neighboring blocks in both horizontal and vertical direction because their motions are strongly correlated. We call this model as 2-D local motion model. In such case, Eq. (22) and (23) can be simplified as

$$v_{i,x}(m,n,i) = a_{100}v_{i,x}(m-1,n,i) + a_{-110}v_{i,x}(m+1,n-1,i) + a_{010}v_{i,x}(m,n-1,i) + a_{110}v_{i,x}(m-1,n-1,i) + w_{i,x}(m,n,i), \quad (24)$$

$$v_{i,y}(m,n,i) = a_{100}v_{i,y}(m-1,n,i) + a_{-110}v_{i,y}(m+1,n-1,i) + a_{010}v_{i,y}(m,n-1,i) + a_{110}v_{i,y}(m-1,n-1,i) + w_{i,y}(m,n,i). \quad (25)$$

The support of the model mentioned above is depicted in Figure 5.

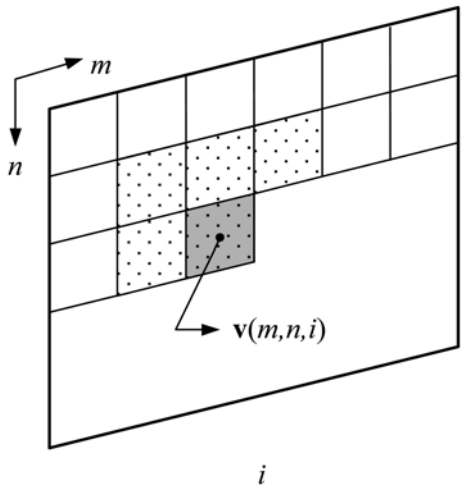


Fig. 5. Causal AR models for motion vector associated with spatial neighboring blocks.

2.3.3 State space representation of spatial local AR model

For the full state propagation, we must represent the proposed models, Eq. (11) and (12), in a state space. Since the Kalman filter is implemented by one-dimensional recursion, it is very difficult to transfer the two-dimensional AR model into one-dimensional state space representation [39,40]. To attack this problem, we introduce an extra deterministic input into the conventional state-space equations, and then we have the state-space representation as follows.

Predicted equation:

$$\mathbf{v}(m,n,i) = \Phi\mathbf{v}(m-1,n,i) + \Lambda\mathbf{u}(m,n,i) + \Gamma\mathbf{w}(m,n,i)\,, \quad (26)$$

where $\mathbf{v}(m,n,i)$ represents the state vector at the location (m,n,i); $\mathbf{u}(m,n,i)$ is the introduced deterministic input; and $\mathbf{\Phi}$, $\mathbf{\Lambda}$, $\mathbf{\Gamma}$ and $\mathbf{H}$ are the corresponding matrices. They are respectively defined as

$$\mathbf{v}(m,n,i)=\begin{bmatrix} v(m,n,i) \\ v(m-1,n,i) \\ v(m+2,n-1,i) \\ v(m+1,n-1,i) \\ v(m,n-1,i) \end{bmatrix}, \mathbf{v}(m-1,n,i)=\begin{bmatrix} v(m-1,n,i) \\ v(m-2,n,i) \\ v(m+1,n-1,i) \\ v(m,n-1,i) \\ v(m-1,n-1,i) \end{bmatrix},$$

$$\mathbf{u}(m,n,i)=\hat{v}(m+2,n-1,i), \quad \mathbf{w}(m,n)=\begin{bmatrix} w(m,n,i) \\ w(m+2,n-1,i) \end{bmatrix},$$

$$\mathbf{\Phi}=\begin{bmatrix} a_{100} & 0 & a_{-110} & a_{010} & a_{110} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \mathbf{\Lambda}=\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \text{and } \mathbf{\Gamma}=\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Measurement equation:

$$\mathrm{z}(m,n,i)=\mathbf{H}\ \mathbf{v}(m,n,i)+e(m,n,i), \tag{27}$$

where $\mathbf{H}=\begin{bmatrix}1 & 0 & 0 & 0 & 0\end{bmatrix}$.

Because the element $v(m+2,n-1,i)$ of the motion vector $\mathbf{v}(m,n,i)$ can not be written in terms of its previous state, here we use the most recent estimate to approximate it, i.e.,

$$v(m+2,n-1,i)=\hat{v}(m+2,n-1,i)+w(m+2,n-1,i). \tag{28}$$

The above equations indicate that the best available estimate is the most recent update of the MV, which is available at time (m,n,i). Through the above process, the motion estimation based on 2-D AR model can be realized by 1-D recursive manner.

2.3.4 Temporal causal AR models for MV

Using the similar definition of the above spatial model, the AR models in the temporal direction are defined as

$$v_x(m,n,i)=\sum\sum\sum_{(k,l,p)\in S^{\oplus}} a_{klp}\, v_x(m-k,n-l,i-p)+w_x(m,n,i), \tag{29}$$

$$v_y(m,n,i)=\sum\sum\sum_{(k,l,p)\in S^{\oplus}} a_{klp}\, v_y(m-k,n-l,i-p)+w_y(m,n,i), \tag{30}$$

where $S^{\oplus}=\{|l|\le 1,\ |k|\le 1,\ p=1\}$. Like the spatial local model, only the adjacent neighboring blocks are considered as model support, as shown in Figure 6. In such case, the state-space representation of Eq. (29) and (30) are

Predicted equation:

$$v(m,n,i) = a_{001}v(m,n,i-1) + \mathbf{\Lambda}\begin{bmatrix} v(m+1,n-1,i-1) \\ v(m,n-1,i-1) \\ v(m-1,n-1,i-1) \\ v(m+1,n,i-1) \\ v(m-1,n,i-1) \\ v(m+1,n+1,i-1) \\ v(m,n+1,i-1) \\ v(m-1,n+1,i-1) \end{bmatrix} + w(m,n,i)\,, \tag{31}$$

where $\mathbf{\Lambda} = \begin{bmatrix} a_{-111} & a_{011} & a_{111} & a_{-101} & a_{101} & a_{-1-11} & a_{0-11} & a_{1-11} \end{bmatrix}$.

Measurement equation:

$$z(m,n,i) = v(m,n,i) + e(m,n,i)\,. \tag{32}$$

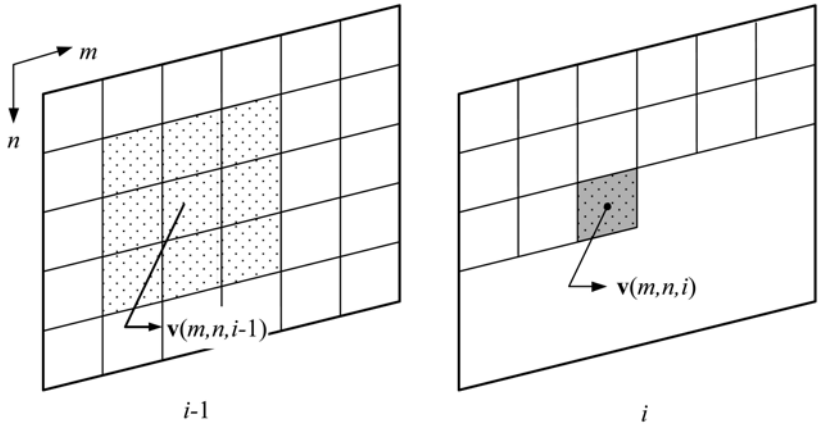


Fig. 6. Causal AR models for motion vector associated with temporal neighboring blocks.

Once the motion models and their state space representation are available, same procedure can then be obtained as in Section 2.3.1.

2.4 Adaptive Kalman filtering

In general, the motion correlation between the adjacent blocks cannot be modeled exactly. Similarly, the measurement of motion vector may have error due to incorrect, inaccurate and low precision estimation algorithms. Therefore, there exist uncertainties in both prediction and measurement processes. The uncertainties of prediction and measurement are represented by zero mean white Gaussian noise $\mathbf{w}(m,n,i)$ and $\mathbf{e}(m,n,i)$ with variance $\mathbf{q}(m,n,i)$ and $\mathbf{r}(m,n,i)$, respectively. In Kalman filtering algorithm, the Kalman gain depends on $\mathbf{q}(m,n,i)$ and $\mathbf{r}(m,n,i)$; therefore, the variances will determine the relative amount of updating using prediction or measurement information [41]. Due to the nonstationary nature of motion vector fields, the values of the variances $\mathbf{q}(m,n,i)$ and $\mathbf{r}(m,n,i)$ should be adjusted block by block to achieve better performance.

In [37], we introduced a distortion function, D_1 and D_2, to measure the uncertainty for both prediction and measurement, and use the distortion function as a reliability criterion. Based on such concept, we calculate the covariance $\mathbf{q}(m,n,i)$ and $\mathbf{r}(m,n,i)$ that are closely related to D_1 and D_2, and then obtain a time-varying Kalman gain. This results in more reliable estimate of MV. The idea behind the procedure is that we use the actual distortion of prediction and measurement to adjust the covariance instead of the conventional complex statistical on-line approaches [37]. Because the distortion measured is more trustworthy than any assumption, the developed scheme achieves very good performance as demonstrated in [37]. The major disadvantages of the scheme are: (i) it needs to send extra side-information, (ii) it increases the overall bit rate, and (iii) the bit stream may be not compatible with the current video coding standard. To overcome this problem, we will introduce an adaptive scheme, which is simpler and more effective than the previous schemes and it does not need to send extra side-information.

We first calculate the errors compensated by predicted MV and measured MV. And then investigate the relation between the difference of the two errors, Δd , and the difference of two motion vectors, ΔMV . Let D_1 and D_2 be the block distortion of motion compensation due to the measurement error and prediction error, respectively, which are defined as

$$\begin{aligned} D_1 &= \frac{1}{M \times N}\sum_{j=0}^{N-1}\sum_{l=0}^{N-1}\left|B\left(m+j,n+l,i\right)-\tilde{B}\left(m+j+z(m,n,i),n+l+z(m,n,i),i-1\right)\right| \\ &= MAD\left(z_x,z_y\right) \end{aligned}, \tag{33}$$

and

$$\begin{aligned} D_2 &= \frac{1}{M \times N}\sum_{j=0}^{N-1}\sum_{l=0}^{N-1}\left|B\left(m+j,n+l,i\right)-\tilde{B}\left(m+j+\hat{v}_x^-(m,n,i),n+l+\hat{v}_y^-(m,n,i),i-1\right)\right| \\ &= MAD\left(\hat{v}_x^-,\hat{v}_y^-\right) \end{aligned}, \tag{34}$$

where B_i and B_{i-1} are the current block and motion compensated block, respectively. The Δd is defined as

$$\Delta d = |\ D_1\text{-}D_2\ |. \tag{35}$$

When ΔMV increases, Δd first increased approximately exponentially, and then decreased exponentially; the increasing rate is larger than the decreasing rate. In general, the measurement is obtained by a real matching; thus a large ΔMV means that the prediction is far away from optimal location. This results in a large value of Δd . However, when ΔMV exceeds a certain value, the measurement may find incorrect position due to the restrictions of block matching, such as cover/uncover-background, complex motion types, etc. Hence Δd will decrease gradually according to the increase of ΔMV . Therefore, we can use two exponential functions to model the variance of prediction as:

$$q(m,n,i)=\begin{cases}1-a_1\exp(-b_1\left\|\mathbf{z}(m,n,i)-\hat{\mathbf{v}}^-(m,n,i)\right\|), & \left\|\mathbf{z}-\hat{\mathbf{v}}^-\right\|\le th\\ a_2\exp(-b_2(\left\|\mathbf{z}(m,n,i)-\hat{\mathbf{v}}^-(m,n,i)\right\|-th)), & \left\|\mathbf{z}-\hat{\mathbf{v}}^-\right\|> th\end{cases} \tag{36}$$

$$r(m,n,i)=1\text{-}q(m,n,i), \tag{37}$$

where *th* is the turning point, which is a reliable index of measurement. If ΔMV is less than *th,* the measurement is reliable compared with prediction. However, when ΔMV is far away from *th*, the measurement is less reliable and the prediction should give more contribution. The parameters *a* and *b* affect the shape of the exponential function and are related with searching methods. The parameter values for full search are larger than those for fast search. Because the prediction can be calculated in the receiver, no extra-information needs to be sent. Therefore, this method is also suitable for real-time application.

2.5 Simulation results

Several image sequences including "Miss America", "Salesman", "Flower Garden" and "Susie" are evaluated to compare the performances of different motion estimation algorithms. The first two sequences are typical videoconference situations. In order to create larger motion displacement, each of the two sequences is reduced to 15 Hz with frame skipping. The last two sequences contain more complex motion such as rotation, and covered/uncovered background. They are converted from CCIR 601 format using the decimation filters recommended by the ISO/MPEG standard committee. The 30 successive frames of each sequence are used in simulation.

Four different algorithms are compared: (i) full search algorithm (FSA), (ii) new three-step algorithm (NTSS), (iii) NTSS combined with 3-D Kalman filter (3DLKF), and (iv) NTSS combined with adaptive Kalman filter (3DALKF). The size of the image block is 16×16. The search window is 15×15 pixels (i.e., S = 7) for "Miss America", "Salesman" and "Susie", 31×31(S = 15) for "Flower Garden". The threshold for the motion detection is 2 for each algorithm. The model parameters are obtained by off-line least-squared estimate. In our work, the parameters are given by c_{100}=7/C, c_{-110}=2/C, c_{010}=7/C, c_{110}=2/C, c_{001}=5/C, c_{-111}=0.25/C, c_{011}=0.5/C, c_{111}=0.25/C, c_{-101}=0.5/C, c_{101}=0.5/C, c_{-1-11}=0.25/C, c_{0-11}=0.5/C, and c_{1-11}=0.25/C. Where C is a normalization factor, and C=26 in our simulation. For non-adaptive algorithm, the covariance of $\mathbf{w}(m,n,i)$ and $\mathbf{e}(m,n,i)$ should be given *a priori*. In this work, the $q(m+2,n-1,i)$ and $r(m+1,n,i-1)$ are 0.095, $q(m,n,i)$ and $r(m,n,i)$ are 0.85 and 0.15, respectively. In the adaptive algorithm, $q(m,n,i)$ and $r(m,n,i)$ are adjusted automatically, the parameters *a*, *b* and *th* are set as a_1=0.55, a_2=1.10, b_1=0.985, b_2=0.009 and th=5.8 for "Flower Garden", a_1=1.10, a_2=0.98, b_1=0.735, b_2=0.008 and th=4.2 for others sequences. The value is obtained experimentally. The $q(m+2,n-1,i)$ and $r(m+1,n,i-1)$ are the same as the non-adaptive algorithm.

The motion-compensated prediction frame is obtained by displacing the previous reconstructed blocks using the estimated motion vectors. Since the estimated motion vector is a real value instead of an integer, the displaced pixels may not be on the sampling grid. Therefore, the well-known bilinear interpolation [17] is adopted to generate a motion compensated prediction frame.

Table 1 summarizes the comparison of the average PSNRs for various algorithms. It indicates that all our algorithms perform better than NTSS. The 3DLKF also obtains better performance than FSA on the average. It is noted that the 3DLKF needs few additional computations over NTSS. The 3DALKF give much better PSNR performance than FSA.

Figures 7-10 displays the comparison of PSNR of the test sequences obtained by various algorithms. It indicates that the proposed method improves the performance. The most

important point to note is that the adaptive algorithm can compensate poor measurement and thereby raise the PSNR significantly. In addition, the visual quality of the reconstructed image is also improved considerably. This can be seen from Figure 11, which shows the reconstructed images of frame 74 obtained by NTSS and 3DALKF, respectively. The NTSS algorithm yields the obvious distortion on some regions such as the left ear and the mouth, as shown in Figure 11 (a). The 3DLKF algorithm, as shown in Figure 11 (c), improves this significantly.

Image Sequence	Algorithm			
	NTSS	FSA	3DLKF	3DALKF
Miss America	38.2581	38.3956	38.6473	38.9077
Salesmen	34.6905	34.7827	34.9477	35.1080
Susie	37.8381	37.8742	38.2893	38.5298
Flower Garden	28.2516	28.4485	28.3836	28.5340
Average	34.7596	34.8753	35.067	35.2699

Table 1. Average PSNR for various algorithms

Figure 12 shows the motion vector fields of "Miss America" obtained by FSA, NTSS and 3DALKF, respectively. The motion vector fields obtained by 3DLKF algorithm are obviously smoother than those by the other algorithms. Although the hierarchical search algorithm presented in [17] can also achieve smooth motion vector fields, it obtains lower PSNR than FSA.

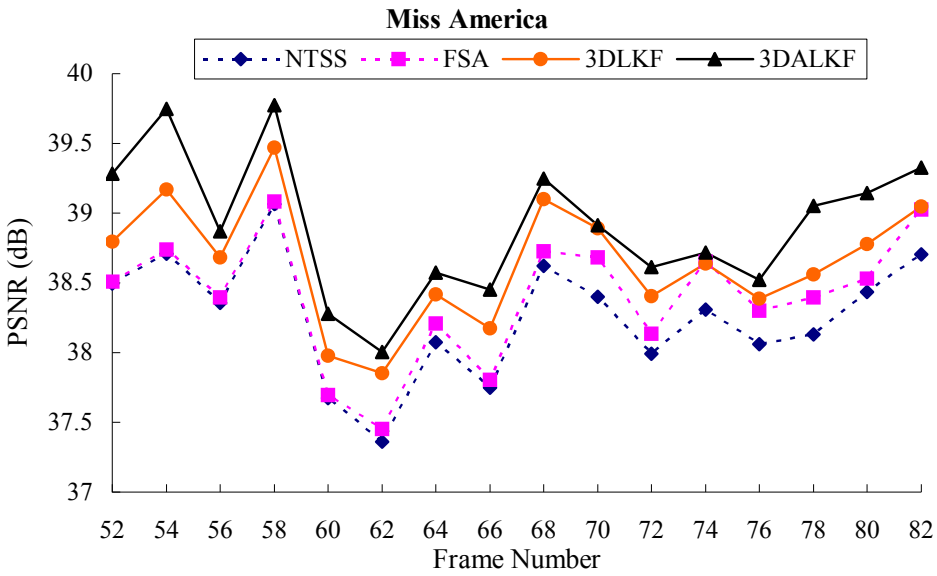


Fig. 7. The PSNR comparison for Miss America sequence at 15Hz

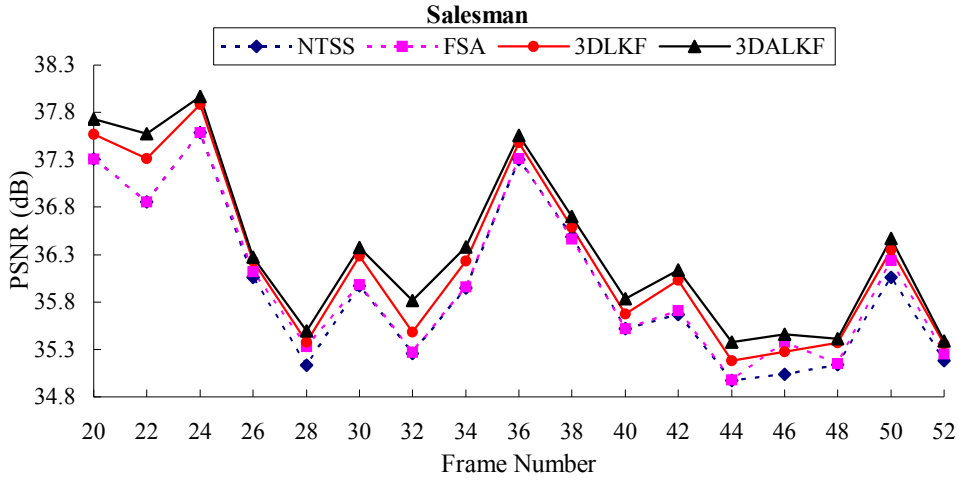


Fig. 8. The PSNR comparison for Salesman sequence at 15Hz

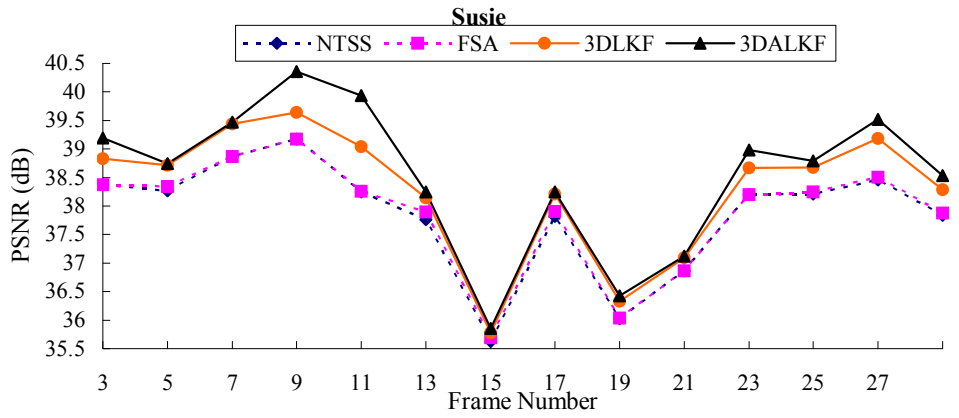


Fig. 9. The PSNR comparison for Susie sequence at 15Hz

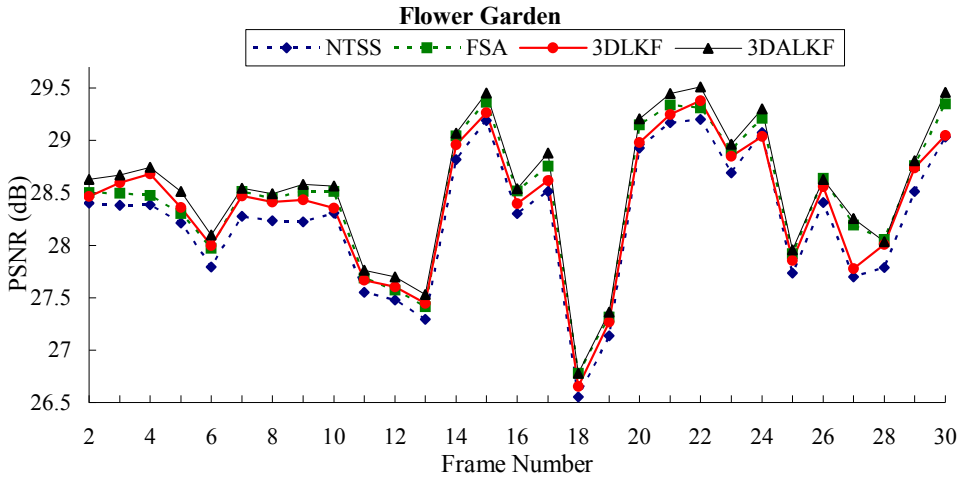


Fig. 10. The PSNR comparison for Flower Garden sequence at 30Hz

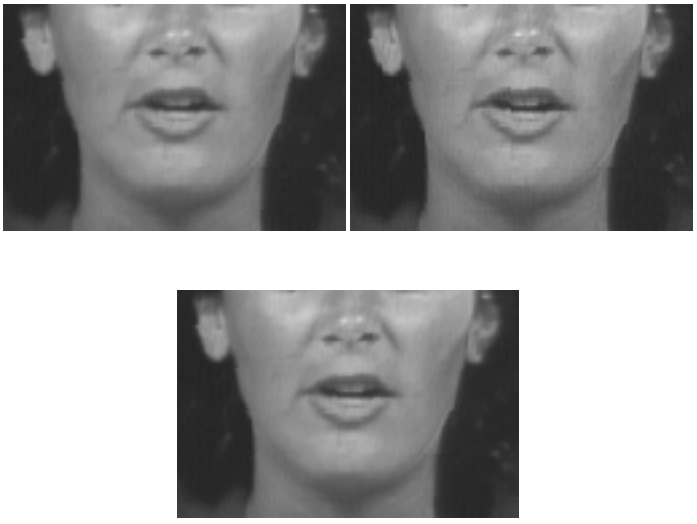


Fig. 11. The comparison of reconstructed image (a) Original image (b) NTSS and (c) 3DALKF.