2006 IEEE International Symposium on Signal Processing and Information Technology

Adaptive Block Motion Prediction

Abouzar Eslami
Department of Electrical Engineering
Sharif University of Technology
Azadi Ave., Tehran, Iran.
Email: eslami@mehr.sharif.edu
Telephone: (+9821) 77950135

Massoud Babaeizadeh
Department of Electrical Engineering
Sharif University of Technology
Tehran, Iran
Email: mbzadeh@yahoo.com

***Abstract*— Block motion estimation is an important field of video processing. This paper presents a new scheme to reach in faster block motion estimation based on motion prediction. The scheme exploits an adaptive filter to predict the block motion from its spatio–temporal motion compensated adjacent blocks. the predicted motion determines the initial candidate block for search methods with biased search center. in special case of partial distortion search, this strategy reduces search time by preventing from complete distortion calculation for more loser candidates. after comparison of three conventional adaptive filters, a normalized least mean square filter with convergence monitoring is recommended for motion prediction from 5 neighbor blocks. Experimental results imply on adaptive filter ability for block motion prediction and its efficiency in reducing the time cost of motion estimation search strategy.**

I. Introduction

Motion estimation is a fundamental process of video coding regarding to its efficiency in reducing the temporal redundancy. Among different approaches for motion estimation, block motion estimation (BME) is especially considered because of its simplicity and efficiency. In this approach the target frame is partitioned to equal size blocks (rectangles). Block motion is assumed as relative distance between corresponding blocks in target and anchor frames. Finding the most similar block in anchor frame performs by exploring a predefined search window and comparing the similarity criterion (sum of square error or SSE):

$$SSE = \sum_{\mathbf{x}\in\mathbf{X}} |I_q(\mathbf{x}) - I_{q-1}(\mathbf{x}-\mathbf{m})|^2. \qquad (1)$$

where $\mathbf{m} = [m^x, m^y]^T$ is the motion vector and $\mathbf{X}$ the set of target block pixels with coordination of $\mathbf{x} = [x, y]^T$. The optimal solution is the result of exhaustive search over all candidates in search window (full search or FS). Although full search results in optimum solution of BME, but its computation cost keeps it away from implementation.

In order to reduce FS cost, two groups of motion estimation algorithms are presented in literature. In the first group the number of candidates is reduced and searching performs on a restricted number of candidates determined by the search pattern [1]–[3]. Approaches of second group do not omit any candidate before taking part in similarity competition (Partial distortion search or PDS for instance) [4]–[7]. These search strategies are based on more efficiently calculation of similarity criterion. The similarity criteria (equation 1) can be implemented by accumulating of partial distortions between the target block and candidate block. Since the partial distortion is non–negative, for a new candidate, iterations of calculating the SSE terminates when the accumulated distortion is larger than minimum distortion pertaining to one previously assessed candidate.

Whatever is the motion estimation method (except FS), a valuable approach to speed up the process is predicting the block motion and bias the center of search pattern to the predicted location of best candidate. Although this scheme can also be effective for methods of first group (TSS and etc.) but its most explicit effect is on PDS and similar methods in which the number of candidates is much more. Regarding to the PDS strategy, the worst situation is when new candidate is better than all previous ones that means all iterations of calculating distortion should be performed. In this situation there is no privilege for PDS. The best situation for PDS is when the first candidate is the best one and PDS never completes iterations. Block motion prediction can propose an appropriate candidate to be exploited as the first candidate in PDS.

Chung et. al. employed adaptive block motion and combined it with three step search and showed its efficiency in reducing search time [9]. Their prediction was a non-linear conditional prediction in which motion of each block was predicted with respect to their neighbors in same frame. Their method can just predict the motion of those blocks whose up and left neighbors' amount of motion are identical. For these few blocks the motion in x direction is set as the m^x of right neighbor and the motion in y direction set as the m^y of upper neighbor. Despite restricted accuracy of this approach but it is fast and easy to implement.

In this paper, an adaptive method is proposed to predict block motion from temporal and spatial neighbors. The motion vector of undergoing block is predicted using a weighted sum of adjacent blocks' motion (previously estimated motion vectors). Extra processes are employed to suppress the effect of those adjacent blocks in present frame whose motion is independent from undergoing block. Furthermore, since the block partitioning is similar in all frames but their inner object moves, processes are necessary to compensate this difference. Indeed the neighbors of block in previous frames are those neighbors after motion compensation and block motion should

0-7803-9754-1/06/$20.00©2006 IEEE

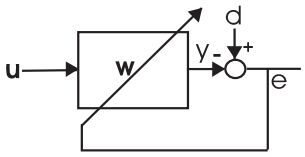


Fig. 1. Conventional structure of adaptive filter

be predicted from neighboring blocks in previous motion compensated frames. With respect to the fact that image sequences does not have stationary conditions, prediction weights should change adaptively. Variation of object velocity and also motion model parameters by time are the most important reasons for updating prediction weights. Various known schemes (employed in adaptive filtering) are applied to update spatial and temporal weights.

The paper is organized as follows. First, in section II, the intra and inter mode block motion predictions are introduced with their theoretic bases. In section III, alternative adaptive filter (AF) schemes are applied for weight update and the figure of acquired quantitative results are depicted to show the validity of BMP in different sequences and difference of AF methods. Section IV combines the adaptive BMP with PDS for constructing fast optimum block motion estimation. Proposed adaptive predictions are applied to various image sequences and section V contains the experimental results and quantitative criterion. Finally, section VI gives an overview on the proposed algorithm and concludes the paper.

II. Motion Vector Prediction

A. inter–mode prediction

Generally, object motion varies smoothly by time. With an appropriate frame rate object motion in target frame correlates with object motion in anchor frame. From basic dynamics, temporal difference of object motion is the result of object acceleration. This acceleration equals to the force divided by object mass. Suppose that m_i and a_i are object velocity and acceleration in frame i respectively, then

$$m_i - m_{i-1} = a_i. \tag{2}$$

With a constant acceleration assumption, equation 2 implies that there is no need for motion estimation (except for two initial frames) and object motion can be exactly predicted from two previous frames. Practically, object acceleration varies by time. Therefore motion prediction is indeed an acceleration prediction problem. Choosing an auto regressive model for object acceleration, its motion is also an AR process.

$$\begin{aligned} a_i &= \alpha_1 a_{i-1} + \alpha_2 a_{i-2} + \cdots \\ m_i &= (1+\alpha_1) m_{i-1} + (\alpha_2 - \alpha_1) m_{i-2} + \cdots \end{aligned} \tag{3}$$

Assuming AR model and employing adaptive filter (AF) structure (figure 4), object motion can be predicted by defining:

$$y = \mathbf{w}^h \mathbf{u} \tag{4}$$

$$\mathbf{u} = [m_{i-1}, m_{i-2}, m_{i-3}, \cdots]^T \tag{5}$$

$$d = m_i \tag{6}$$

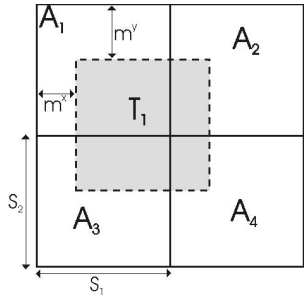


Fig. 2. Inter–mode motion compensation

It must be mentioned that $m = m^x + jm^y$, $\mathbf{u}$, $\mathbf{w}$ and d are complex and adaptive filter calculation should perform in complex domain. The AR model can be appropriate for real objects' motion. In the real world, object acceleration is a smooth time function since sudden change in acceleration needs large amount of force. Experiments show that this assumption is fairly acceptable for objects with mass. For an object with negligible mass (lips and eyes for instance), object motion goes away from AR model and approaches to white noise. The AR model of object motion is also addressed in [8] and some of object motion prediction literature but has never been applied to BME.

In BME, frames are partitioned by predefined blocks and blocks do not move by object movement. Therefore the filter input vector ($\mathbf{u}$) can not be simply defined as previous block's motion. Objects can appear and disappear in one block through frames and block motion history includes different objects' motion. Suppose that block $[n_1, n_2]^T$ by the size of $[s_1, s_2]$ had a motion $m = m^x + jm^y$ in previous frame. In present frame, the block center locates on $[n_1 \times s_1 + m^x, n_2 \times s_2 + m^y]^T$ and probably, the addressed block replaces with block $[\hat{n}_1, \hat{n}_2]^T = [n_1 + m^x/s_1, n_2 + m^y/s_2]^T$. Consequently, motion of block $[\hat{n}_1, \hat{n}_2]$ should be predicted from motion of block $[n_1, n_2]$ in previous frame. Modifying the AR model of object motion, each block motion can be linearly predicted from blocks' motion of previous motion compensated frames. Actually, after motion compensation, two temporal adjacent blocks pertain to same part of one object and the AF based BMP can be acquired reconstructing $\mathbf{u}$.

Usually, block motion is not a correct multiple of block size and transmitted block does not completely replace another one. Refer to figure 2, where block A_1 (up–left) transmitted as $m^x < s_1$ and $m^y < s_2$. Transmitted block T_1 can overlap with up to four blocks. Its motion distribute over four blocks and changes their first previous motion $\mathbf{u}_i\{1\}$ with respect to their overlapping surface. Therefore, for $i = 1, 2, 3$ and 4

$$Re\{\mathbf{u}_i\{1\}\} = \mathcal{S}\{T_1 \cap A_i\} \times m^x \tag{7}$$

$$Im\{\mathbf{u}_i\{1\}\} = \mathcal{S}\{T_1 \cap A_i\} \times m^y \tag{8}$$

where $\mathcal{S}\{T_1 \cap A_i\}$ is the common surface of translated block

(T_1) and block A_i.

$$\mathcal{S}\{T_1 \cap A_i\} = \begin{cases} \frac{(s_1-m^x)(s_2-m^y)}{s_1 s_2}, & \text{i=1;} \\ \frac{m^x(s_2-m^y)}{s_1 s_2}, & \text{i=2;} \\ \frac{(s_1-m^x)m^y}{s_1 s_2}, & \text{i=3;} \\ \frac{m^x m^y}{s_1 s_2}, & \text{i=4.} \end{cases} \quad (9)$$

Totally, each block's first element of input vector ($\mathbf{u}\{1\}$) is sum of all block's motion in previous frame weighted by corresponding surface overlap ratio. There is a simple extension for large motions with $m^x > s_1$ or $m^y > s_2$ by replacing m^x and m^y in equations 9 with $mod(m^x, s_1)$ and $mod(m^x, s_1)$ respectively. For more histories of block motion in **u**, just more motions should be saved. Consider that there is no need for extra calculations since for each block, just the value of ($\mathbf{u}\{1\}$) should be computed from equation 9. Indeed, the older motions can be resulted from a shift in input vector (**u**).

B. Intra–mode prediction

In spite of inter–mode BMP efficiency, it is useless in scene changes when there is a small correlation between anchor and target frames. In addition to temporally adjacent blocks, spatially neighbors can also be employed for BMP. Intra–mode prediction is based on correlation between each block and its neighbors in same frame. When moving object is enough big and blocks reasonably small, it will divide to some blocks and motion of these blocks correlate since they all pertain to one object. In specific application of BMP, only already processed blocks can participate in prediction. The restriction rises from the fact that the only available motions in motion input vector (**u**) are those already estimated motions of present frame.

A simple prediction is to first check if two adjacent blocks belong to one object and then assign the neighbor's motion (previously estimated) to undergoing block [9]. This prediction is fast but limits the type of object motion to translation. The other discussing point is choosing criterion for detecting blocks of same object. equality of up and left blocks' motion, employed in [9], is not true always. Especially when upper and left blocks belong to one object but the undergoing block does not. Even if upper and left blocks have identical sum of motion vectors, it is not correct to assign motion of these blocks to the undergoing block regarding to motion model type.

From motion modelling, pixels of object with affine movement displace with

$$\begin{cases} m^x = a_0 + a_1 x + a_2 y \\ m^y = b_0 + b_1 x + b_2 y \end{cases} \quad (10)$$

$$\Rightarrow \begin{cases} m^x(x', y') = m^x(x, y) + a_1 \Delta x + a_2 \Delta y \\ m^y(x', y') = m^y(x, y) + b_1 \Delta x + b_2 \Delta y \end{cases} \quad (11)$$

and in complex notation of motion:

$$m(x', y') = m(x, y) + (a_1 + jb_1)\Delta x + (a_2 + jb_2)\Delta y \quad (12)$$

In the case of translation, all model parameters except a_1 and a_2 are zero and motion of neighboring blocks are identical (assumption made in [9]). But in more general motion models as affine these parameters can have any value and should be approximated for more precise prediction. Although only a and b coefficients are unknown and need approximation but equation 12 is held just when adjacent blocks belong to one object. The more general equation for implementing by AF structure is:

$$m(x', y') = \alpha m(x, y) + (a_1 + jb_1)\Delta x + (a_2 + jb_2)\Delta y \quad (13)$$

Ideally, α is a binary weight which is set to 1 if pixels $[x', y']^T$ and $[x, y]^T$ both belong to same object, and set to zero otherwise. Binary weights are not compatible with AF update strategy, so α should be a continues variable with hope that α in extreme condition, converge to one of 0 and 1 values and prepare a decision whether two blocks are independent or not. The alternative, employed in this study, is initializing and fixing α with respect to the optical flow, difference of intensity and other acceptable benchmarks. These benchmarks can probably determine two adjacent blocks belong to one object or not. Assuming that the objects are rigid and do not break apart, if two adjacent blocks belong to one object, they preserve this property in future frames. All these assumptions can be correct when blocks move with objects. So the strategy employed for modifying the previous motion vectors in inter–mode is also necessary here. Mention that equation 13 represents prediction for pixels and in the case of blocks $[x, y]^T$ replaces with $[n_1, n_2]^T$ and $[\Delta x, \Delta y]^T$ with block size. Consequently, the intra–mode BMP can be implemented with AF structure by defining $\mathbf{u} = [m_n, s_1, s_2]^T$ where m_n is the neighbor's estimated motion. Same as inter–mode prediction, d is desired block motion ($d = m^x + jm^y$) and **w** is a complex weight vector. But $\mathbf{w}\{1\}$ is binary, predefined and constant and **u** contains additional elements as block size (s_1 and s_2).

For other motion models as bilinear, more additional elements (as $s_1 s_2$) will be involved since pixel displacement in these models are function of xy, x^2, y^2 and etc. While bilinear model is theoretically more general than affine but increases the number of adaptive weights. Large number of weights introduces a problem namely over training in *neural network*. The weights will minimize error in adaptation but do not prepare good prediction for new frames. On the other hand since s_1 and s_2 are constant, $s_1 s_2$ and other polynomial combinations of them are also constant and can be omitted since the adaptive weight of s_1 and s_2 can exactly cover them ($ws_1 s_2 = (ws_1)s_2 = \hat{w}s_2$). There is a similar condition for s_1 and s_2. Both of them are constant and w can continuously vary, so they are nothing but a multiple of each other. Thus it is not necessary to have both of them in **u** and it can simplify to $\mathbf{u} = [m_n, 1]^T$.

From equation 13, if motion of undergoing block is predicted from its upper block then $\Delta x = 0$ and in the case of left block $\Delta y = 0$. However if all of the first neighbors (up, left and up–left) are involved then the motion of undergoing

block ($[m^x, m^y]^T$) can be computed from:

$$m^x = m_l^x + a_1^l \Delta x + m_u^x + a_2^u \Delta y + m_{ul}^x + a_1^{ul} \Delta x + a_2^{ul} \Delta y \quad (14)$$

$$m^y = m_l^y + b_1^l \Delta x + m_l^y + b_2^u \Delta y + m_{ul}^y + b_1^{ul} \Delta x + b_2^{ul} \Delta y \quad (15)$$

Based on previous discussions, all the terms of a^l, a^u , a^{ul} and associating b weights can merge in one constant and complex weight. Totally, in order to predict block motion from first nearest neighbors by AF structure, the input vector $\mathbf{u}$ should be defined as $\mathbf{u} = [m_u, m_l, m_{ul}, 1]^T$. Three first elements of the weight vector are binary and constant determined after inspecting independence of corresponding blocks with undergoing block. The last weight is a complex variable weight and updates frame by frame.

III. Block Motion Prediction with Adaptive Filter

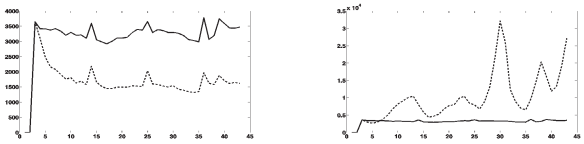
Since introduction of adaptive filter, it is widely employed in different applications as system identification, beamforming and etc [10]. Its adaptation is completely desirable in non–stationary conditions and signals. An special application of adaptive filter is linear prediction and in this study, block motion prediction is defined as special AF based linear prediction. The sturctuer of an adaptive filter and the required vectors are already introduced. It must be mentioned that block motion is predicted from both spatial and temporal neighbors. The input vector is the combination of those in intra– and inter–mode predictions. When the number of reference frames is two (as in this study) and three first nearest neighbors are exploited or prediction, the input vector and weight vectors have 6 elements. Different adaptive filters can reach to different PSNR accuracy regarding to their update strategy and adaptation ability. Without an accurate prediction, total motion estimation time increases. Therefore a fast AF is not acceptable if it can not adapt with input sequence. Such an adaptive filter can even degrade the PDS. In this study, three adaptive filters are inspected in both time consumption and prediction quality.

Least Mean Square is the first and almost most frequently addressed approach. Minimizing the statistical cost function $J = E\{|e - \mathbf{w}^h \mathbf{u}|\}$, results in wiener filter $\mathbf{w}_o = R^{-1}\mathbf{P}$. Where $R = E\{\mathbf{u}^h \mathbf{u}\}$, $\mathbf{P} = E\{\mathbf{u}^h d^*\}$ and $E\{.\}$ represents statistical expectation. Exploiting wiener filter in lack of these two statistical quantities is impossible. Such limitation leads to a suboptimal but more applicable solution of AF namely least mean squares (LMS). This adaptive filter is completely similar to the steepest descent implementation of wiener filter. Except that the $\mathbf{R}$ and $\mathbf{P}$ are replaced with $\mathbf{u}^h\mathbf{u}$ and $\mathbf{u}d^*$ respectively. More details on LMS can be found in [10] and here just the weight update equation in iteration q and its correction coefficient (μ) condition are addressed.

$$\mathbf{w}[q] = \mathbf{w}[q-1] + \mu \mathbf{u}[q] e^*[q] \quad (16)$$

$$0 < \mu < 2/\lambda_{max} \quad (17)$$

where λ_{max} is the largest eigen value of autocorrelation matrix. LMS is an efficient adaptive filter in noise cancellation regarding to its simplicity and quality. An important issue in LMS is appropriate selection of μ. Crucial big values of μ result in filter divergence. As it can be inferred from fig. 3, an increase in μ from 0.016 to 0.022 can result in filter divergence and rapid increase in prediction error. On the other side, small μ increases convergence time and also mean of prediction error (fig. 3).



(a) (b)

Fig. 3. LMS sensitivity to μ in this application. prediction error of *flower garden* sequence. Dashed lines are prediction error while solid line is sum of frame motion amplitude per frame, (a) $\mu = 0.016$, (b) $\mu = 0.022$.

Weight update amount of LMS in (eq. 17) depends on input dynamic range. In order to remove this disadvantage, in normalized LMS, the weigh update and convergence condition are replaced with:

$$\mathbf{w}[q] = \mathbf{w}[q-1] + \frac{\mu}{\|\mathbf{u}\|^2} \mathbf{u}[q] e^*[q] \quad (18)$$

$$0 < \mu < 2 \quad (19)$$

Selection of μ is easier in NLMS and is not a function of filter input energy. But the prediction AF is still susceptible to μ value. NLMS is more recommended when the input signal has wide dynamic range while the motion vector has a restricted dynamic range. A typical value for μ is 0.8. more details are available in [10].

Against these two adaptive filters which exploit statistical gradient, the third adaptive approach, namely recursive least square error (RLS), uses sum of real error value.

$$J = \sum |e(i) - \mathbf{w}(i)^h \mathbf{u}(i)|^2 \quad (20)$$

omitting the details of RLS and its implementation (available in [10]), the basic relations of RLS are:

$$y = \mathbf{w}^h[q-1]\mathbf{u}[q] \quad (21)$$

$$\xi = d[q] - y \quad (22)$$

$$\mathbf{k} = \frac{\lambda^{-1} p[q-1]\mathbf{u}[q]}{1 + \lambda^{-1}\mathbf{u}^h[q]p[q-1]\mathbf{u}[q]} \quad (23)$$

$$\mathbf{w}[q] = \mathbf{w}[q-1] + \mathbf{k}\xi^* \quad (24)$$

$$p[q] = \lambda^{-1} p[q-1] - \lambda^{-1}\mathbf{k}\mathbf{u}^h[q]p[q-1] \quad (25)$$

Comparing RLS equations with those of LMS, RLS, they have more cost both in calculation and memory. For each block a matrix of p in addition to filter weights ($\mathbf{w}$) should be saved while in LMS only a $\mathbf{w}$ vector needs memory. Multiplying by the number of blocks in each frame, the difference between two schemes in memory requirements reveals.

IV. Proposed block motion estimation using prediction

Predicting block motion and exploiting it as first candidate of PDS reduces time cost of search. The AF produces the

predicted motion by weighted summation of spatial and temporal neighbors' motion. However, starting the search from predicted first candidate, PDS finally extracts the optimum motion required for weight update in AF. Each block has its own AF which updates per frame exactly.

Among three AF schemes, complexity increases from LMS to RLS while the prediction quality enhances a little (refer to the experimental results). The most outstanding advantage of RLS is its convergence while LMS has a strict sensitivity to correction coefficient (μ). But computation complexity and memory requirement is too high. Such process cost maybe acceptable in computer based applications but it is hardly affordable in many others. Regarding to insensitivity of NLMS to the input vector energy, it can be recommended for BMP with an experimental μ in the range of $[0.5 - 0.7]$. Generally a small μ results in late convergence and usually more average prediction error. But a large μ can make AF diverge. Basically, if the predicted motion is worse than zero initial motion (conventional PDS) it takes more time for PDS to find optimum candidate. Such a situation is probable when AF diverges. So divergence should be prevented either by small μ or monitoring of prediction error. It seems that small μ is not desirable since video sequence is non–stationary and small μ can not follow its variations.

The alternative is choosing bigger μ to speed up adaptation and then check total error on whole frame and compare with simple threshold. If AF was going to diverge the filter weights reset to zero (conventional PDS) and filter update restarts with a smaller or even same μ. Totaly, proposed PDS with adaptive BMP can be implemented in following steps:

- for each target block:
  - predict block motion.
  - perform PDS with predicted motion as the first candidate.
  - update filter weights.
- modify the filter input vectors $\mathbf{u}$ of all blocks.
- check convergence.

The AF based BMP application does not restrict to PDS. It can also be used in partial ridgelet distortion (PRDS) [7], NPDS [5]and even in suboptimal motion estimation methods as TSS [1], FSS [2], HS [3] and etc. All of these schemes can perform faster if their first candidate is the total search winner. Proposed modification on PDS is independent from previous speed up schemes. It attempts to predict the best candidate (the optimum result of PDS) as the first one and consequently sooner termination of distortion calculation. It is recommended to combine this with other schemes as pixel priority in [6]. This will increase looser rejection probability too.

It is also possible to extract pixel priority by an AF structure. Furthermore, a better candidate can be expected by increasing filter order. Employing AF structure for pixel priority detection costs too much in calculation and time. Such AF needs many weights and converges late, while it is possible for block and inside object completely disappear before convergence. Fuzzy membership functions are more appropriate candidates for block pixel priority detection. Including more temporal neighbors may result in better prediction since it gives a better approximate of acceleration time function while this means more memory requirement. But, generally increasing AF order is not efficient. Experimental results show that increasing filter order, increases convergence time and prediction error. Regarding to non–stationary property of the image sequence, the AF should converge as fast as possible. However, regarding to the relatively small number of blocks in each frame, memory requirement is not crucial (especially in computer implementation).

V. Experimental Results

Proposed PDS is applied to three different benchmark sequences. Since predicted block motion is just used for extracting the first candidate, the total search error is identical for conventional PDS and the proposed one. Two important criteria of proposed methods are search time and prediction accuracy. Table I contains motion estimation time cost per frame. As this table shows, if AF divergence be prevented, adaptive block motion prediction (ABMP) can reduce the search time by the cost of memory and more complexity. Rate of enhancement completely depends on undergoing sequence. Prediction is highly suitable for sequences with camera movement as *flower garden* and is fairly applicable to outdoor sequences. But in the case of close up and facial sequences where fine objects have rapid and random motion, prediction fails (*car phone*). However preventing from divergence guarantees that search time in worth conditions does not increase too much, while usually decreases.

In order to inspect the ability of different adaptive filtering methods in BME, figures 4–6 depict prediction error per frame for different sequences. Dashed lines are prediction error while solid line is sum of frame motion amplitude per frame. The value of μ in LMS is set to 0.01, and set to 0.8 in NLMS. Against what was expected, the acquired results are not far better, except that there is no more sensitivity to update parameter (such as μ in LMS). Comparing the prediction error of different sequences in tables II and III, it can be inferred that there is not a good prediction in *car phone* sequence, because of small and non–smooth motions of face and body. On the other side, in *flower garden* and *mobile* linear prediction can perform successfully. Against what was expected from AF theory, RLS was not far better predictor. May be the employed LMS reaches too close to the minimum available error. However it seems that paying the cost of RLS is not justifiable if extra processes for monitoring of AF convergence are provided.

While the method of [9] do not require adaptation and is simpler for implementation, ABMP usually prepares a better prediction of block motion since it employs both spatial and temporal neighbors. ABMP covers more motion models in comparison with Chung's method in which just translation is mentioned.

TABLE I

SEARCH TIME OF PDS WITH AND WITHOUT ADAPTIVE BLOCK MOTION PREDICTION

	mobile	*flower garden*	*car phone*	*tennis*
PDS with ABMP	1808.41	586.04	907.5	623.92
conventional PDS	2016.52	657.07	890.47	645.09

TABLE II

MEAN OF PREDICTION ERROR (DIFFERENCE OF PREDICTED AND ESTIMATED MOTIONS).

	mobile	*flower garden*	*car phone*	*tennis*
LMS	1.01	0.74	0.69	0.61
NLMS	0.92	0.68	0.76	0.580
RLS	0.90	0.67	0.76	0.55
method of [9]	1.56	0.76	0.94	0.83

TABLE III

VARIANCE OF PREDICTION ERROR

	mobile	*flower garden*	*car phone*	*tennis*
LMS	4.28	2.17	2.14	2.01
NLMS	2.53	1.78	2.02	1.64
RLS	2.18	1.70	1.97	1.54
method of [9]	5.86	1.78	1.83	2.19

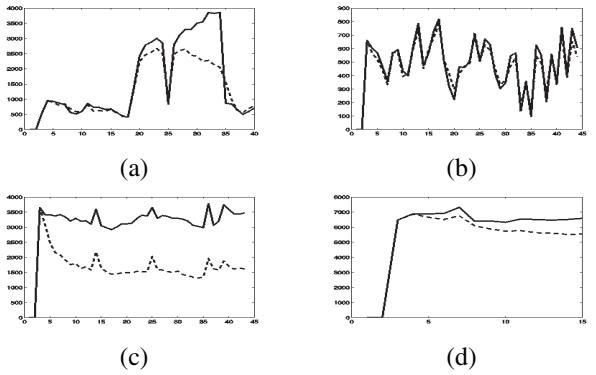


Fig. 4. The prediction error of applying LMS (dashed line) and sum of frame motion (solid line): (a) *tennis* sequence, (b) *car phone*, (c) *flower garden*, (d) *mobile*.

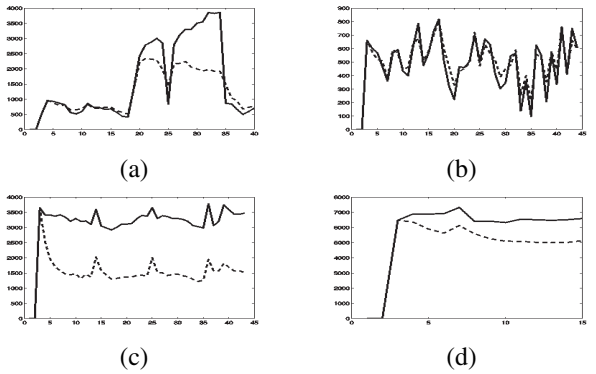


Fig. 5. The prediction error of applying NLMS (dashed line) and sum of frame motion (solid line): (a) *tennis* sequence, (b) *car phone*, (c) *flower garden*, (d) *mobile*.

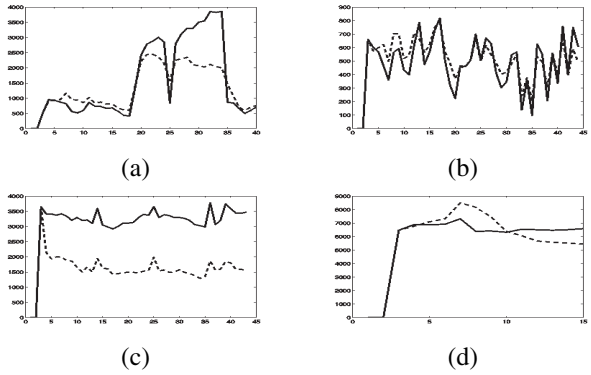


Fig. 6. The prediction error of applying RLS (dashed line) and sum of frame motion (solid line): (a) *tennis* sequence, (b) *car phone*, (c) *flower garden*, (d) *mobile*.

VI. CONCLUSION

This paper presents a new scheme to reach in faster block motion estimation based on motion prediction. Exploiting an adaptive filter structure, the block motion is predicted from its spatio–temporal motion compensated blocks. Different adaptive filters are discussed and experimentally assessed. The final approach is based on NLMS with convergence monitoring. The predicted motion by NLMS from 5 neighbors is used for determining the first candidate of search algorithm. This scheme increases search speed by preventing from complete distortion calculation for losers. This block motion prediction can combine with many other search strategies to reach in even faster algorithms. Experimental results imply on adaptive filter ability for block motion prediction and BMP efficiency in reducing the time cost of motion estimation search strategy.

REFERENCES

[1] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion Compensated Interframe Coding for Video Conferencing," in Proc. Nat. Telecommun. Conf. New Orleans, LA, Nov. 1981, pp. G5.3.1-G5.3.5.

[2] L.M. Po and W.C. Ma, "A Novel Four-Step Search Algorithm for Fast Block Motion Estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 6, pp. 313-317, June 1996.

[3] C. Zhu, X. Lin, and L. P. Chau, "Hexagon-based search pattern for fast block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 12, pp. 349-355, May 2002.

[4] C. Bei and R. M. Gray, An improvement of the minimum distortion encoding algorithm for vector quantization, IEEE Trans. Commun., vol. COM-33, pp. 11321133, Oct. 1985.

[5] C. K. Cheung and L. M. Po, "Normalized partial distortion search algorithm for block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 10, pp. 417-422, Apr. 2000.

[6] Y. Chan, K.C. Hui, and W.C. Siu "Adaptive Partial Distortion Search for Block Motion Estimation," ElSEVIER Journal of Vis. Commun. Image R., Vol. 15, pp. 489-506, 2004.

[7] M.Eslami, E.fatemizadeh, "Edge Sensitive Block Motion Estimation Employing Partial Ridgelet Distortion Search", submitted to ISSPIT2006.

[8] A. Elnagar, A.M.Hussein, "an adaptime motion prediction model for trajectory planner systems", Proceedings of IEEE international Conf. on robotics and automation, pp. 2442-2447, 2003.

[9] H.Y. Chung, P.Y.S.cheung and N.H.C. Neung, "Adaptive Search Center Non-linear Three Step Search", Proceedings of IEEE ICIP 98.

[10] S. Haykin, "Adaptive Filter Theory," Prentice Hall, 3. rd. Edition,