Chen, X., Canagarajah, C. N., Nunez-Yanez, J. L., & Vitulli, R. (2012). Lossless video compression based on backward adaptive pixel-based fast motion estimation. Signal Processing: Image Communication, 27(9), 961–972. 10.1016/j.image.2012.06.004

Lossless Video Compression based on Backward Adaptive Pixel-based Fast Motion Estimation

Xiaolin Chen[a,*], Nishan Canagarajah[a], Jose L. Nunez-Yanez[a], Raffaele Vitulli[b]

[a]*Department of Electrical and Electronic Engineering, University of Bristol, Bristol, BS8 1UB, UK*
[b]*Eupropean Space Agency (ESA), On-Board Payload Data Processing Section, The Netherlands*

Abstract

This paper presents a lossless video compression system based on a novel Backward Adaptive pixel-based fast Predictive Motion Estimation (BAPME). Unlike the widely used block-matching motion estimation techniques, this scheme predicts the motion on a pixel-by-pixel basis by comparing a group of past observed pixels between two adjacent frames, eliminating the need of transmitting side information. Combined with prediction and a fast search technique, the proposed algorithm achieves better entropy results and significant reduction in computation than pixel-based full search for a set of standard test sequences. Experimental results also show that BAPME outperforms block-based full search in terms of speed and entropy. We also provide the sub-pixel version of BAPME as well as integrate BAPME in a complete lossless video compression system. The experimental results are superior to selected state-of-the-art schemes.

Keywords: lossless compression, video compression, pixel-based prediction, motion estimation, fast search.

*Corresponding author
Email addresses: `patlincxl@gmail.com` (Xiaolin Chen), `Nishan.Canagarajah@bristol.ac.uk` (Nishan Canagarajah), `J.L.Nunez-Yanez@bristol.ac.uk` (Jose L. Nunez-Yanez), `Raffaele.Vitulli@esa.int` (Raffaele Vitulli)

1. Introduction

Recently, lossless video compression has become more and more important, especially for emerging applications such as video archiving, digital cinema, remote sensing, medical imaging etc. [1]. Similar with the lossy counterpart, lossless video compression has been studied in both transform and spatial domain. In transform domain, Oami and Ohta [2] proposed to approximate the Discrete Cosine Transform (DCT) with an integer-matrix transform and performed a mapping-based quantization process, to achieve lossless capability. Also, various forms of 2-D [3, 4] and 3-D [5] wavelet transforms were investigated in the context of lossless video compression. In spatial domain, motion estimation is usually applied to remove the temporal redundancy. Due to the shift-variant property [3, 6] and the added complexity of operating in the transform domain, we are in favor of performing motion estimation in the spatial domain.

There are generally two kinds of motion estimation: block-matching algorithm (BMA) and pixel-based algorithm (PBA). BMA is simple and efficient. A large number of researchers employed and improved BMA in their lossless video compression schemes [7, 8, 9, 10, 11]. Memon and Sayood were among the pioneers looking into the problem of lossless compression of video [7]. They presented a hybrid compression scheme that adaptively switches between the temporal predictor based on BMA and the spectral predictor which takes the best predictor among spectral planes. Brunello et al. [8] introduced a temporal prediction scheme based on motion estimation and optimal 3-D linear prediction. Carotti et al. [11] combined the CALIC [12] framework with a temporal predictor which uses multi-frame motion estimation and least square method to weigh predicted values from multiple frames. The advanced video compression standard H.264/MPEG-4 AVC [13], featuring multi-frame variable block-size block-base motion estimation, has the advantage of being flexible and efficient. It has a 4x4 and a 16x16 block-based intraframe prediction, each of which supports 9 and 4 prediction modes respectively. Its "High Profile" includes a simple predictive method to encode the marcoblocks losslessly. These BMA based methods need to transmit the motion vectors (MV) to the decoder. When high accuracy of motion estimation is required, the size of MV often increases and thus becomes a large burden. Pixel-based algorithm (PBA), on the other hand, makes use of the past information so as to avoid sending large amount of MV as side information. Yang et al. [14] proposed a simple pixel-based scheme which adaptively chooses either intra-frame or inter-frame predictor by comparing the variance of the intra-frame and inter-frame neighbourhood. Its inter-frame predictor directly takes the pixel in the previous frame with the same position of the current pixel as predicted value. Carotti et al. [1, 15] used the previous two frames to predict the pixel in the current frame. It is based on the assumption that the motion registered between the previous two frames continues at the present time. Li et al. [16] designed a temporal predictor which used a local neighbourhood to do full search (FS) within the search area and finds the one that minimizes the mean of absolute difference (MAD). It also includes wavelet transform and spatial prediction in the video compression system. These pixel-based motion estimation schemes need to perform at least a certain amount of motion estimation on the decoder side as well as the encoder.

However, much as it is efficient in temporal decorrelation, motion estimation is computationally intensive and can consume up to 80% of the total computational requirement in the H.264/AVC encoder (JM software) [17, 18, 19]. Therefore many *fast search* techniques have been invented to alleviate the heavy computation of full search, which exhaustively evaluates all possible positions within the search area in the reference frame. In the popular block-matching algorithm, full search is assumed to be able to achieve the best motion estimation accuracy, regardless of the bitrate, due to its exhaustiveness which, however, leads to very heavy computation load. The idea of fast search is to examine only a few positions instead of all positions in the search area. This would often result in less accurate estimation of the motion than using full search. The goal of fast search is to reduce the computation as much as possible while maintaining the motion estimation accuracy. Classical representative fast search examples are three step search [20], diamond search [21, 22] and hexagonal search [23].

While full search and fast search have been widely used in block-matching motion estimation [13, 8], only full search is adopted by pixel-based schemes [16]. Fig. 1 shows the main trend of algorithms in literature (solid line). One of our contributions, as the dash line shown in Fig. 1, is to bridge pixel-based algorithms and fast search, on which we are not aware of any research work in literature. This well serves our goal to develop a lossless video compression system that is able to achieve high compression ratio with reduced complexity. Our previous letter [24] has introduced the concept of a backward adaptive pixel-based fast predictive motion estimation (BAPME) for this purpose. In this paper, we will provide the first complete descrip-

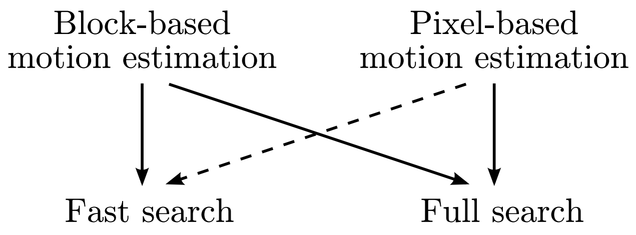


Figure 1: Main trend of algorithms in literature vs. our approach (dash line).

tion of BAPME and discuss more details of it in depth. We also provide the sub-pixel version of BAPME, as well as a complete lossless video compression system using BAPME. We have to emphasize that the techniques proposed here mainly aim for lossless video compression and hence we are not considering the rate-distortion problem.

This paper is organized as follows. In Section 2, we propose a pixel-based fast predictive motion estimation and discuss its performance and complexity. The sub-pixel version of the motion estimation and its empirical performance are presented in Section 3, followed by the complete system and its performance in Section 4. Conclusions are drawn in Section 5.

2. Pixel-based Fast Predictive Motion Estimation

In this section, we describe the Backward Adaptive pixel-based fast Predictive Motion Estimation (BAPME) scheme and discuss various issues of it. The goal of BAPME is, without transmitting any overheads, to obtain smaller residual after motion estimation and to reduce the computational complexity. Because no motion vector is transmitted, BAPME is a symmetric coding scheme, where the whole procedure has to be performed at both the coder and the decoder. In brief, the main novelties of BAPME are: 1) it is pixel-based; 2) it includes a new prediction scheme for initial motion vector; 3) it performs fast search on our designed routine. We explain these features in detail below.

1. The proposed motion estimation scheme is conducted on a pixel-by-pixel basis, due to the natural difficulties of block-matching: first, objects in video usually have irregular shapes. Some schemes introduced variable block size (from 16×16 to 4×4) to adapt to the video contents [13, 18], but this still has limitation in shapes and increases the

complexity and the amount of side information; second, the rigid motion (e.g. rotation and zoom) and nonrigid motion (e.g. elastic or deformable motion) are hard to model by rigid blocks; third, the transmission of side information is undesirable. To this end, we seek an alternative pixel-based approach which cares for each individual pixel and possibly achieves better motion estimation accuracy without sending any side information. In contrast with BMA, our pixel-based scheme is more "prediction-based" rather than "matching-based", because the pixel to be coded is not included in any matching operations and the fast search criterion only serves as a reference for prediction.

2. BAPME works in two stages. Firstly, it predicts an initial motion vector (MV) using a new prediction scheme. We study the characteristics of MVs and carefully design a MV prediction scheme consisting of four predictors. Note that we still use the term "motion vector" here although we do not send them as side information.
3. In the second stage, starting from the initial MV obtained from the previous stage, BAPME employs a target window, which is a neighbourhood of the current pixel, to search within a search range in the previous frame and locates the MV that minimises the sum of absolute difference (SAD) of the target window. This search is conducted on a fast diamond search pattern [21, 22]. The fast search technique is normally applied in BMA, but we use it in PBA to reduce the computational complexity and to increase the prediction accuracy.

We explain the technical details of the two stages below.

2.1. Initial Motion Vector Prediction

The first stage of the proposed scheme is initial motion vector prediction. MV prediction has been used to reduce search steps in BMA [19, 17]. Since this initial motion vector is used as the starting point of the following fast search, it does not only to speed up the overall search but also to make better prediction of the motion and to reduce the likelihood of MV being trapped in a suboptimal point, which occurs sometimes when an MV with minimal SAD but irrelevant content is found.

There are some analyses on the empirical average MV distribution for BMA in literature [25, 26]. The experiments were carried out by using full search with SAD as the block distortion criterion. It was revealed that for the Rec.601 RGB standard test sequences, within a search range of 31×31, about

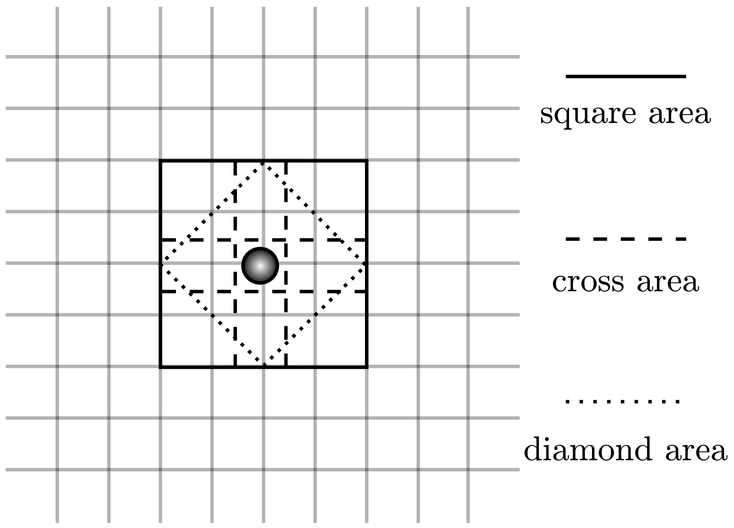
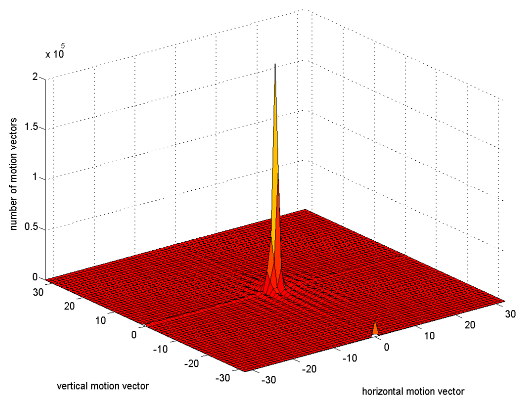
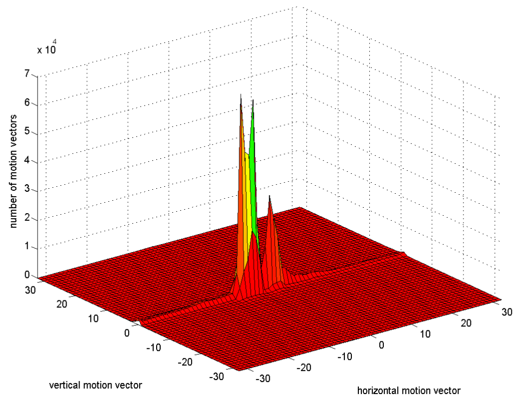


Figure 2: Search areas: cross, diamond and square region (pixels are denoted on grid crossings).

23% MVs were at zero $(0, 0)$, 47% in the cross region, 48% in the diamond region and 52% in the square region within an absolute distance of 2 pixels, as shown in Fig. 2. For CIF and QCIF sequences, within a search area of 15×15, more than 74% and 90% MVs concentrate in the cross region, and over 81% and 93% MVs are within the 5×5 square region. In our own experiments, full search using 4×4 block size within a commonly used search range of ± 32 is applied on the G plane of the RGB sequence "football" and on the Y plane of the YUV sequence "bus". The resulting MV distributions are displayed in Fig. 3. From both figures, apart from the high peaks at the centre, there are lower bumps along the cross regions. This means that the MVs are highly peaked at the centre (0,0) and the cross regions. Fig. 3b indicates that the MV distribution of "bus" concentrates on one direction because the sequence "bus" shows two buses running horizontally with different speed. In Fig. 3a, there is a small peak far away from the centre, which happens when there is abrupt changes in the scene. Another way to observe the behaviour of MV is through the MV field. A MV field is an image showing the direction and length of the MV of each block by arrows. The MV field of two frames in the CIF sequence "foreman", shown in Fig. 4, illustrates that a large part of the video is static or slow-moving and the MVs are mostly correlated, either in terms of length or direction.



(a) Motion vectors distribution of RGB video "football"



(b) Motion vectors distribution of YUV video "bus"

Figure 3: Motion vectors distribution obtained using full search and 4×4 block size within a search range of ± 32 pixels.

Although the above analyses and observations are done by performing block-based search, they still provide a good reference of the behavior of the MVs given that block-based search can reasonably discover the true motion. Therefore, we can summarise some characteristics of motion and motion vectors:

- Most of the motion in video sequences at standard frame rate (30f/s) is slow and smooth.
- Motion vectors often concentrate in centre area due to stillness and in cross area due to simple translation;
- Generally, there exists more horizontal motion than vertical motion in natural video sequences, e.g. car running as in Fig. 3b or human walking etc.
- It might be beneficial to use a large search area when the motion activity is high. However, it is difficult to classify the motion magnitude, as it can vary a lot in a few frames or even within a frame. Moreover, enlarging the search area will significantly increase computation and side information for BMA full search.

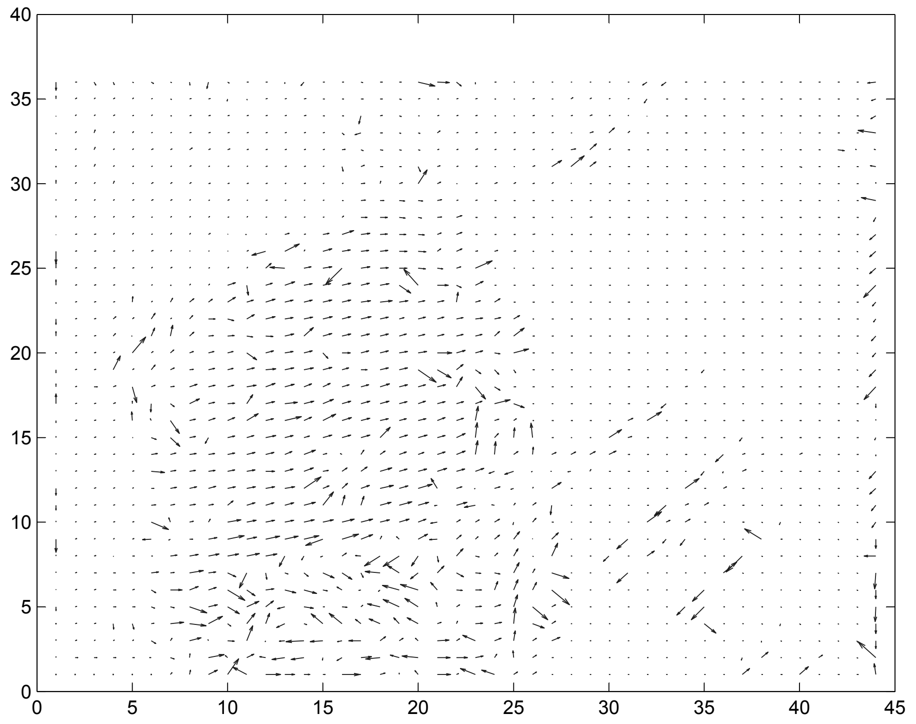


Figure 4: Motion vector field of two frames in sequence "foreman". The arrows indicate the direction and length of the motion vector of each 8×8 block.

- There is often high correlation among neighbouring MVs, indicating smooth motion. This is not always true as in Fig. 4 that there exist some "outliers" in the motion field. It is because sometimes noise exists or lighting condition changes, or the block-matching method is incapable of dealing with complicated motion.

Motivated by these motion features, we propose to use a set of four motion vector predictors – west predictor, neighbour predictor, median predictor and centre predictor ($\{\vec{MV}_w, \vec{MV}_{nei}, \vec{MV}_m, \vec{MV}_0\}$). Among them, the neighbour predictor is newly designed and the other three have been introduced for BMA in the literature. We assume here that each frame is scanned in the raster order and pixels above and to the left of the current pixel are known.

1. West predictor $\vec{MV}_w$. Based on the above analyses and a commonly agreed conclusion that the movement in the horizontal direction is much

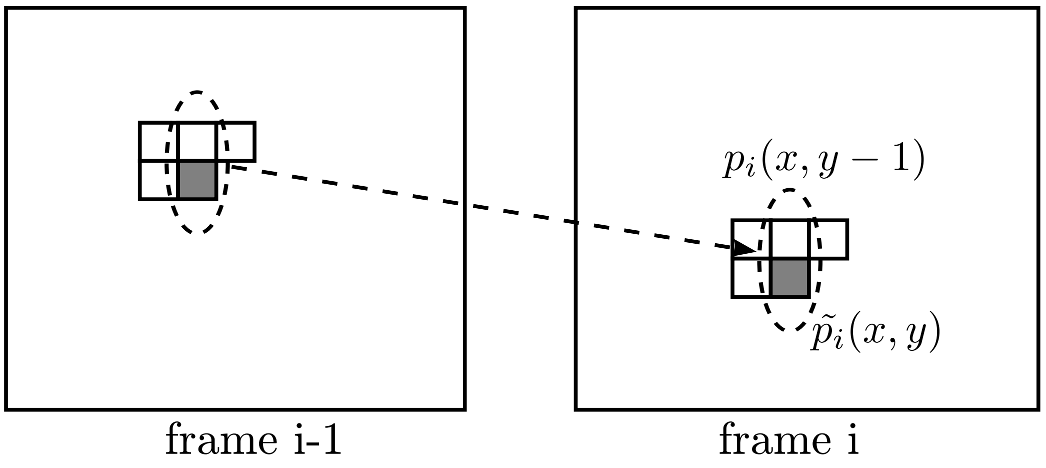


Figure 5: Neighbour predictor for initial MV prediction of pixel $p_i(x, y)$. The MV of $p_i(x, y-1)$ is chosen if $\tilde{p}_i(x, y) \approx p_i(x, y-1)$, see Section 2.1.

heavier than that in the vertical direction for natural video sequences [17], we take the MV of the pixel to the west of the current pixel as one of the predictors.

2. Neighbour predictor $\vec{MV}_{nei}$. We construct this new predictor to make use of the spatial correlation among pixels for motion estimation. It is based on the assumption that if the current pixel belongs to the same object as any one of its known neighbouring pixels, they are very likely to move together from the previous frame, or in other words, be in the same relative position in the previous frame. As illustrated in Fig. 5, for example, if the pixel $p_i(x, y-1)$ is identified as being in the same group with the current pixel $p_i(x, y)$ in the current frame i, we predict that $p_i(x, y)$ moves together with pixel $p_i(x, y-1)$. So the MV of pixel $p_i(x, y-1)$ in the current frame is chosen as the predictor of the MV of the current pixel. However, to define which neighbouring pixel is in the same group with the current pixel is not an easy task. A possible solution is segmentation, but its complexity might outweigh the benefit that an initial MV prediction can bring. Instead, we observed that pixels in the same group are generally connected and have similar pixel intensity. Therefore, we decide whether an adjacent pixel (west, northwest, north and northeast to the current pixel) is a "pair" with the current pixel by comparing their intensity, and the one with smallest difference is chosen. Since the value of the current pixel is unknown in

a backward adaptive scheme, we consider that it is reasonable to make a good prediction of the current pixel $p_i(x, y)$ and use the predicted value $\tilde{p_i}(x, y)$ for the comparison. The Gradient-Adjusted Prediction (GAP) from CALIC [12] is simple and effective in spatial prediction, and hence we adopt it here to predict the current pixel. Note that using this prediction does not put more burden on the computational complexity since there are often areas where temporal prediction does not perform well and spatial prediction is considered as a very helpful alternative in a robust video coding system. We will describe our proposed complete video compression system in Section 4.

3. Median predictor $\vec{MV}_m$. It is used due to the nature of the smooth and continuous movement in most of the sequences. Median predictor calculates the median value of the MVs of the pixels to the west, north and northeast of the current pixel.
4. Centre predictor $\vec{MV}_0$. The above analyses show that there often exist a large static area in video and MV is often centre-biased, so we include $(0, 0)$ as an initial motion vector predictor.

From the four initial MV candidates, we choose the initial MV with the minimum SAD in a target window.

$$\vec{MV}_{init}(m, n) = argmin_{(m,n)} SAD(m, n) \tag{1}$$

$$SAD(m, n) = \sum_{(x,y)\in tw} |p_i(x, y) - p_{i-1}(x + m, y + n)| \tag{2}$$

where (m, n) is the initial MV candidates, tw is the target window, and $p_i(x, y)$ and $p_{i-1}(x, y)$, $(x, y) \in tw$ denote the pixels within the target window in current frame i and previous frame $i-1$, respectively. We will be discussed the selection of target window in the next subsection.

Table. 1 shows the empirical average probability of these initial MV being the final chosen MV after motion estimation and the probability of none of them being the final one. If the initial MV is exactly the same as the final one, the search steps in the next stage are reduced to only 2 (large and small DS once each), as will be demonstrated in the next subsection. If the probability of the predictor is high, we regard the predictor as good because: a) it gives good prediction for $\vec{MV}_{init}$; b) it saves search steps in the next stage. Among the four predictors, the west predictor $\vec{MV}_w$ has the highest probability, and the newly designed neighbour predictor $\vec{MV}_{nei}$ comes second. There are

Table 1: Probability of initial MV being the final MV (in %).

sequence	$\vec{MV}_w$	$\vec{MV}_{nei}$	$\vec{MV}_m$	$\vec{MV}_0$	none of them
Football	68.37	61.01	56.60	19.53	20.54
Mobile	67.61	57.33	55.66	4.41	20.77
Susie	61.51	51.47	45.94	7.72	25.82
Missa	54.74	40.86	36.86	8.35	31.62
Claire	72.90	66.68	55.41	44.74	16.19
Average	65.03	55.47	50.09	16.95	22.99

Table 2: Probability of chosen initial MV being the final MV (in %).

sequence	$\vec{MV}_0$	$\vec{MV}_m$	$\vec{MV}_{nei}$	$\vec{MV}_w$
Football	18.95	39.09	10.45	5.61
Mobile	4.21	52.16	8.34	9.64
Susie	8.34	39.79	12.01	9.08
Missa	8.26	30.63	11.24	13.59
Claire	53.19	15.10	7.15	3.92
Average	18.59	35.35	9.84	8.37

about 23% final MVs that are different from any of the initial predicted MVs. The values on each row do not sum up to 100%, because some of the MVs from different predictors are the same, e.g. sometimes $\vec{MV}_w = \vec{MV}_m$.

At this point, it is also interesting to know how these predictors are distinct from each other. We show the probability of the chosen initial MV being the final MV in Table 2. Different from Table 1, these values show the distinction among different predictors and there is no overlap among predictors. The percentage of each predictor does not include the amount that is overlapped with the previously examined predictors, so the order of examining predictors does affect these values. The order of examining predictors in this table is $\{\vec{MV}_0, \vec{MV}_m, \vec{MV}_{nei}, \vec{MV}_w\}$. In terms of the effect of the newly introduced neighbor predictor, in our experiment there are around 7% of it being chosen as the initial vector while being different from other predictors.

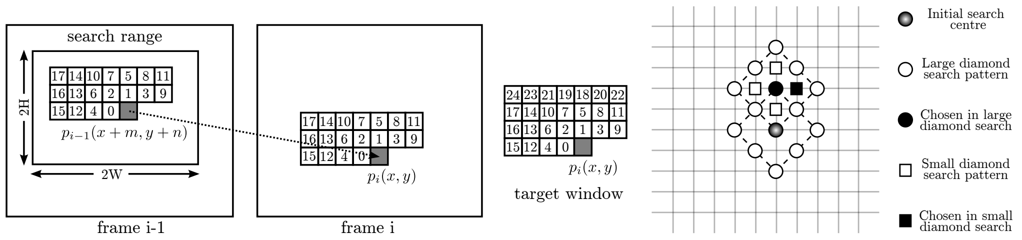
This table cannot show the usefulness of each predictor quantitatively, but it does provide evidence that each predictor contributes to the prediction.

Given the results in Table 1 and Table 2, we would like to keep the whole prediction set to improve the robustness of the scheme, especially when dealing with more complex motion. To keep the computation cost minimum, when two MV candidates are the same, the SAD only needs to be calculated once. The order of checking the MV candidates also affects the result slightly. This is due to the fact that different MVs might give the same SAD, which is also the reason of the second column in Table 2 being slightly different with the fifth column in Table 1. For this reason, changing the order of checking the MV predictors results in slight difference of the probability in Table 1, but it does not affect the probability ranking of predictors in our experiment. So we give higher priority to the predictor that is more likely to predict the final MV according to Table 1. Although the result of Table 1 is obtained with a certain prediction order, in our experiment we also compare other possible orders of arranging the predictors. Therefore, the order is arranged based on this probability from high to low, which is $\{\vec{MV}_w, \vec{MV}_{nei}, \vec{MV}_m, \vec{MV}_0\}$, resulting in minimum amount of search steps in the following fast search. Note that based on the order in Table 1, the amount of MVs being the final MVs are 77% while it is 72.15% based on the order in Table 2. As will be presented in Section 2.3, this prediction stage contributes a 0.06bits per pixel (bpp) reduction to the zero-order entropy of the motion estimation scheme.

2.2. Pixel-based Fast Search

We propose to conduct pixel-based prediction on a fast search pattern in the second stage of our scheme, with the initial motion vector from the first stage as the search centre. This new approach enhances the accuracy of pixel-based prediction.

The pixel-based predictor uses a *target window* to evaluate the motion between the current frame i and previous frame $i-1$. We define the *target window* as upper-left pixels of the current pixel $p_i(x, y)$ in frame i, as in Fig. 6a. The order of the pixels is arranged according to their Euclidean distance to the current pixel. The shape and the size of the target window can be adjusted if needed. Table 3 shows our experiment results using target window size of 25, 18, 15, and 12 on the test sequences. The result indicates that choosing 18 neighbouring pixels is adequate and gives the best results compared to a bigger or smaller target window. For each pixel to be coded, the predictor searches within a search range $2W \times 2H$ in frame $i-1$ for the


search range
2H
2W
$p_{i-1}(x+m, y+n)$
frame i-1
$p_i(x,y)$
frame i
$p_i(x,y)$
target window
Initial search centre
Large diamond search pattern
Chosen in large diamond search
Small diamond search pattern
Chosen in small diamond search

(a) Target windows in the current frame and previous frame, within search range $2W \times 2H$.

(b) Diamond search pattern

Figure 6: Pixel-based adaptive motion estimation, see Sec. 2.2. (a) Target window (b) Diamond Search Pattern

Table 3: Zero-order entropy of BAPME residue using different target window sizes, in bits per pixel(bpp)

sequence / window size	Football	Mobile	Susie	Missa	Claire	Average
25	4.946	**4.456**	3.652	3.662	**2.514**	3.846
18	**4.914**	4.470	**3.636**	**3.661**	2.525	**3.841**
15	4.915	4.480	3.641	3.665	2.535	3.847
12	4.932	4.505	3.657	3.672	2.546	3.862

target window which minimises the SAD in Eq. (4) with the target window in the frame i. Note that in our experiment, there is no direct link between the size of the target window and the search range.

Using SAD as the criterion of selecting the best predictor is due to its simplicity and effectiveness in our experiments. We compare the zero-order entropy of the residue of BAPME using SAD and mean square errors (MSE) as the motion distortion criteria in Table 4. MSE is also a popular choice for measuring motion distortion. It is good at detecting the edges but is more sensitive to outliers. MSE is given by

$$MSE = \frac{1}{N} \sum_{(x,y)\in tw} (p_i(x,y) - p_{i-1}(x+m, y+n))^2 \tag{3}$$

where N is the number of pixels within the target window, which is 18 in our experiment for both criteria. Table 4 shows that SAD gives slightly better results than MSE. In addition to the much less complexity of SAD, we choose SAD for our scheme. Note that the block-based decode side motion

Table 4: Zero-order entropy of BAPME residue using different motion distortion criteria, in bpp.

sequence	Football	Mobile	Susie	Missa	Claire	Average
SAD	4.914	4.470	3.636	3.661	2.525	3.841
MSE	4.929	4.489	3.650	3.674	2.518	3.852

vector estimation/derivation (DMVD) [27] also has a similar approach in terms of defining a target area and using a matching scheme to obtain the motion vector, but BAPME differs in taking advantage of prediction and fast search, which will be described soon. In the end, the current pixel $p_i(x, y)$ is predicted by

$$\hat{p}_i(x, y) = p_{i-1}(x + m_0, y + n_0) \tag{4}$$

where (m_0, n_0) is the final MV for $p_i(x, y)$ in the search range.

We propose applying the above pixel-based predictor on a fast search pattern so that the target window only examines a few search points instead of exhaustively examining every position within the search range. We choose the simple diamond search (DS) pattern for three reasons: a) the previous MV analyses show that a large portion of "best MVs" resides on the cross region, which is covered directly by the DS pattern; b) the large DS pattern covers eight directions with similar step size to ensure a thorough evaluation in various directions; c) it has better accuracy for motion estimation than the hexagonal pattern [23], which is important for lossless compression because a smaller amount of residual can be achieved. Therefore, we perform diamond search following the steps in Fig. 6b. Starting from the initial MV as search centre, the target window in previous frame is checked on the large DS pattern with step size 2 until the position with minimum SAD is found at the centre. Then the target window is checked on a small DS pattern with step size 1. The small DS pattern is only evaluated once and the MV that minimises SAD is chosen as the final MV, denoted as (m_0, n_0) in Eq. (4). The benefit of this method, especially when combined with the initial MV prediction, is not only reducing the complexity, but also providing directional prediction on the object motion. The performance results in the following section demonstrate this. Note that using a simple DS pattern serves for a proof-of-concept purpose here, and future work on applying more sophisticated fast search pattern can possibly improve the efficiency and accuracy.

2.3. Experimental Performance of BAPME

To evaluate the performance of BAPME, we carry out experiments on the green colour component of a set of Rec. 601 standard RGB test sequences. Although our method only applies to the green component in this section, it can be applied in the same way in any other components. Since BAPME is designed for lossless video compression, we are only concerned with the compression ratio. One of the most straightforward and meaningful ways to reveal this is to calculate the entropy of the sequence residue after motion estimation. First, we examine performance of the new prediction and fast search in BAPME by comparing it with other *pixel-based* motion estimation methods in Table 5. Full search(FS), hexagonal-based search (HEXBS) [23] and diamond search (DS) [21, 22] using the same target window as BAPME without prediction are chosen for comparison. This is because: a) full search has been recognized as giving better performance than fast search methods due to it exhaustiveness; b) HEXBS is a fundamental and popular fast search pattern; c) DS is a benchmark fast search motion estimation and is also the fast search pattern that we adopted. The search range is restricted to ± 32. Note that the FS, DS and HEXBS here are pixel-based searching using our target window, so they do not need to send any motion vector and thus the result in Table 5 does not include any side information. Our experiment shows that enlarging the search range does not improve the result for these test sequences. For better understanding the distinction of BAPME from BMA, we also calculate the mean of absolute differences (MAD) of pixels in the target window for different *pixel-based* algorithms in Table 6, with the same setting as the experiment in Table 5. Note that the MAD here does not calculate the difference between the pixel values and its estimate, but rather give the match of the target window and thus cannot directly be compared with the entropy in Table 5.

We highlight the best entropy and the best MAD value for each sequence respectively in Table 5 and Table 6. Table 6 shows that full search (FS) has the best MAD amongst all these motion estimation schemes, which is consistent with the theory and with our common knowledge of BMA. However, Table 5 shows that HEXBS and DS occasionally outperform FS in terms of zero-order entropy, which does not occur in BMA. This important and interesting finding is because BMA FS always obtains the global minimal mean of errors in each block and these errors are directly encoded, which naturally leads to minimum entropy. However, in PBA, although FS still obtains the minimum MAD, as in Table 6, it does not guarantee the best prediction on

Table 5: Zero-order entropy of residue of different *pixel-based* algorithms within search range ± 32, in bpp. Size of the video is in parentheses.

sequence	FS	HEXBS	DS	BAPME
Football (720×486)	**4.91**	5.25	5.05	**4.91**
Mobile (720×576)	4.56	4.60	4.53	**4.47**
Susie (720×480)	3.68	3.82	3.71	**3.64**
Missa (360×288)	**3.66**	3.71	3.68	**3.66**
Claire (360×288)	2.59	**2.52**	**2.52**	**2.52**
Average	3.88	3.98	3.90	**3.84**

Table 6: Average MAD of different *pixel-based* algorithms within search range ± 32.

sequence	FS	HEXBS	DS	BAPME
Football	**4.61**	10.18	7.34	5.85
Mobile	**3.32**	4.24	3.89	3.73
Susie	**1.81**	3.19	2.46	2.16
Missa	**1.68**	2.43	2.23	2.11
Claire	**0.82**	0.98	0.94	0.92
Average	**2.45**	4.20	3.37	2.95

the current pixel. Since the target window in PBA does not include the pixel to be encoded, the position with minimum SAD is not necessarily an optimal point for the current pixel. The target window in a BMA is for "matching", while in PBA it is more for providing information for "prediction". HEXBS and DS can roughly estimate the motion directions and avoid the situations of finding a target window with minimal SAD but non-relevant content, as possibly in FS. Consequently, they can sometimes provide better prediction than FS in this pixel-based setting. Moreover, based on this advantage of DS, BAPME includes a new initial MV predictor which makes use of the intra-frame spatial correlation to help with motion estimation, resulting in improved performance. It achieves lower or equivalent entropy than other schemes on every sequence. On average, it outperforms FS by 0.04bpp, HEXBS by 0.14bpp and DS by 0.06bpp. Compared the performance of DS and BAPME, we can also see the improvement brought by the initial MV prediction. We have to mention that [16] suggests a pixel-based method with possible refined search approach using initial MV and smaller target window.

Table 7: Zero-order entropy of residue comparison of *block-based* FS with different block size and BAPME, within search range ± 32, in bpp.

sequence	16×16	8×8	4×4	BAPME
Football	5.25 (5.27)	4.68 (**4.79**)	4.54 (5.16)	4.91
Mobile	4.54 (4.55)	4.44 (4.53)	4.12 (4.71)	**4.47**
Susie	3.81 (3.84)	3.64 (3.78)	3.22 (3.89)	**3.64**
Missa	3.87 (3.90)	3.65 (3.81)	3.12 (3.83)	**3.66**
Claire	2.56 (2.57)	2.48 (2.57)	2.11 (2.71)	**2.52**
Average	4.01 (4.03)	3.79 (3.91)	3.42 (4.06)	**3.84**

But it is based on full search without prediction and no performance data on their motion estimation is available for comparison.

We are also interested in the performance difference of the pixel-based BAPME and BMA. Therefore, we compare BAPME to *block-based* FS with different block size, since FS works best in BMA in terms of bit rates. The search range of both is ± 32. Table 7 shows the residue entropy of FS excluding side information in the second to fourth columns. It indicates that the zero-order entropy of BAPME is placed between block size 16×16 and 8×8. The uncompressed side information requires $2log_2(2 \times search\_range)/(block\_size)^2$ bpp. We compress it with the state-of-the-art JPEG-LS as in [8], the overall entropy including compressed side information is shown in parentheses. In our experiment BAPME is superior to conventional block-based FS in terms of overall zero-order entropy by a significant margin.

2.4. Complexity Analysis of BAPME

Previous research favours BMA over PBA for video coding, mostly due to the simplicity and speed of BMA. In this section we analyse the complexity of BAPME and demonstrate that pixel-based scheme can also be both effective and efficient. We compare the computational complexity of BAPME and *block-based* FS in Table 8. The number of search points per pixel in BAPME is calculated by

$$SP_{bapme} = SP_{init} + 8 + M \times (S_{ds} - 1) + 4 \qquad (5)$$

where SP_{init} indicates the search points in the initial MV prediction, shown in the second row of Table 8, followed by 8 search points in the first large

Table 8: Computational complexity comparison of BAPME and block-based full search. The units for rows 2-4 are search points, and for rows 5-8 are the absolute difference computation.

sequence	Football	Mobile	Susie	Missa	Claire	Ave.
SP_{init}	2.27	2.48	2.55	2.67	1.93	2.38
S_{ds}	1.16	1.19	1.24	1.31	1.14	1.21
SP_{bapme}	15.01	15.37	15.65	17.48	14.55	15.34
C_{bapme}	270.13	276.62	281.75	289.96	261.96	276.09
$C_{fs}(\pm 8)$	289	289	289	289	289	289
$C_{fs}(\pm 16)$	1089	1089	1089	1089	1089	1089
$C_{fs}(\pm 32)$	4225	4225	4225	4225	4225	4225
SIR(± 8)	7.0	4.5	2.6	-0.3	10.3	4.7
SIR(± 16)	303.1	293.7	286.5	275.6	315.7	294.4
SIR(± 32)	1464.1	1427.4	1399.5	1357.1	1512.9	1430.3

DS pattern. S_{ds} is the search steps in DS. M is either 3 or 5, when the search direction is towards the edge or the corner of the diamond pattern, respectively. On average, there are 80% search steps moving towards the corner and 20% towards the edge [26]. In addition, there are 4 search points in the small DS. The search points per pixel for block-based FS depend on the search range.

$$SP_{fs} = (2 \times search\_range + 1)^2 \tag{6}$$

The amount of absolute difference computation per pixel, shown on the fifth to eighth rows of Table 8, is calculated by

$$C_{bapme} = SP_{bapme} \times tw\_size \tag{7}$$

$$C_{fs} = SP_{fs} \tag{8}$$

From the above equations we obtain the Speed Improvement Rate (SIR) of BAPME over block-based FS. SIR is commonly used to measure the speed improvement [23, 26, 28], given by

$$SIR = \frac{C_{fs} - C_{bapme}}{C_{bapme}} \times 100\% \tag{9}$$

From Table 8, BAPME is slightly faster than a block-based FS scheme when the search range is ±8, but is almost 3 times faster when the search range of

FS is ± 16, and is 14 times faster when the search range is ± 32. Note that the data about BAPME in Table 8 is obtained with search range ± 32. Changing the search range does not affect the computational amount of BAPME too much, since the fast search steps are very limited anyway. For block-based FS, reducing the block size would largely increase the side information, and enlarging the search range would increase both the side information and computation significantly. On the other hand, for block-based fast search, the number of search points are at the same level with our scheme. Because we need to calculate the SAD of the target window for each pixel as in Eq.(7), our scheme requires more computation than the block-based fast search scheme, but with the bonus of more accurate prediction.

3. Sub-pixel Motion Estimation

The above motion estimation is performed with integer pixel as units. However, motion displacement in real world is hardly exactly at integer pixel positions. Although it is easy to use the nearest integer pixel to approximate the fractional pixel position value, improved motion estimation accuracy might be gained if we can actually use the value of the fractional pixel for prediction. By comparing fractional pixel value in motion estimation, a fractional motion vector is obtained. This is called *sub-pixel motion estimation*. Sub-pixel motion estimation has been used in various video coding standards, such as H.264/AVC [13] and VC-1 [29], and has been studied and improved by many researchers [17, 30]. However, it is mostly used in block-matching algorithms. In this section, we will explore the effect of sub-pixel motion estimation on our pixel-based scheme BAPME.

3.1. Methodology

Sub-pixel motion estimation has half pixel (also called half-pel), quarter pixel (also called quarter-pel) and even eighth pixel precision [13]. It is computationally demanding due to the extra search points and interpolation, and especially so for the quarter-pel precision and when complicated interpolation is used [13]. Therefore, we perform BAPME first to obtain the best integer MV, and then search for the interpolated half-pel position immediately adjacent to this position once. If quarter-pel precision is required, the quarter-pel values around the chosen half-pel position are calculated and searched. We use a smaller target window, which is the first seven neighbouring pixels in

Table 9: Zero-order entropy of different pixel-based algorithms with integer-pel and half-pel motion estimation, within search range ± 32, in bpp.

sequence	Football	Mobile	Susie	Missa	Claire	Ave.
FS-IP	4.91	4.56	3.68	3.66	2.59	3.88
FS-HP	4.87	4.50	3.64	3.62	2.54	3.83
DS-IP	5.05	4.53	3.71	3.68	2.52	3.90
DS-HP	4.99	4.41	3.66	3.62	**2.48**	3.83
HEXBS-IP	5.25	4.60	3.82	3.71	2.52	3.98
HEXBS-HP	5.17	4.44	3.76	3.63	**2.48**	3.90
BAPME-IP	4.91	4.47	3.64	3.66	2.52	3.84
BAPME-HP	**4.85**	**4.37**	**3.59**	**3.61**	**2.48**	**3.78**

Fig. 6a, for calculating the SAD at half-pel position. This makes sense because at such a fine scale, only the most adjacent neighbouring pixels give the relevant information for prediction. In our experiments, using a bigger target window did not perform better than a small one. Currently we use the bilinear interpolation to calculate the half-pel values for its simplicity and satisfactory performance, and only the eight sub-pixels surrounding the best integer pixel are searched once. However, integrating the quarter-pel motion estimation is achievable and straightforward.

3.2. Experimental Performance of Sub-pixel Motion Estimation

We present the experimental performance of BAPME with half-pel precision. Table 9 shows the zero-order entropy of the residue after FS, DS, HEXBS and BAPME with integer-pel (IP) and half-pel (HP) precision. Usually, it is not easy to gain a lot of bit saving when the entropy is already reduced to a very low level from previous steps, while it might be relatively easier to improve from a coarser decorrelation scheme because there is likely more redundancy remaining. However, Table 9 indicates that half-pel motion estimation gives more benefit to BAPME than to the pixel-based FS on every sequences. It saves 0.05 bpp for FS and 0.06 bpp for BAPME. Our explanation for this is that BAPME actually finds an integer MV that is closer to the true MV, and hence it is possible to find a better half-pel MV around this integer MV.

To take a closer look at the contribution of half-pel motion estimation to BAPME, we summarise the effective half-pel search ratio and better and

Table 10: Effective half-pel search ratio, and better and worse half-pel ratio(in %).

sequence	Football	Mobile	Susie	Missa	Claire	Ave.
Effective HP	0.76	0.75	0.71	0.68	0.50	0.68
better HP	0.37	0.37	0.31	0.30	0.15	0.30
worse HP	0.27	0.25	0.21	0.20	0.11	0.21

worse half-pel ratios in Table 10. Effective half-pel search ratio measures how often the final half-pel MV being different (either better or worse) from the integer-pel MV chosen from small DS. Better half-pel and worse half-pel ratios measure how often the actual errors given by half-pel MV are smaller or bigger than the ones resulted from integer-pel MV. Table 10 shows that the average effective half-pel search is 68%, out of which about 30% of half-pel are better and 21% are worse. This says that half-pel motion estimation is not always effective on every pixel. Since the half-pel value does not exist and is obtained by interpolation, it might introduce some noise during this operation and hence the worse half-pel. But overall, half-pel motion estimation brings improvement on the result by having more better half-pel.

The above analysis provides evidence that half-pel motion estimation does give reasonable bit rate savings, given that we only use a simple version. We expect more benefit from the sub-pel motion estimation when better interpolation and quarter-pel multiple iteration search is used. Generally, the bit rate saving tends to diminish as the sub-pixel precision increases. The computational complexity also increases significantly as motion estimation goes to finer scale. We can tailor BAPME for a trade off between performance and complexity.

4. Complete Lossless Video Compression System

Based on the proposed motion estimation scheme BAPME, we propose a complete lossless video compression system, named BAPME-VC (BAPME based Video Compression). Motion estimation can effectively predict the pixels when video frames are closely correlated temporally, that is when there is smooth and slow motion between adjacent frames. But it does not always work well when there are abrupt changes or complicated motion in scenes which is difficult to capture by motion estimation. Consequently, a robust video compression scheme normally includes image compression techniques to exploit the intra-frame correlation when the above scenarios happen. For

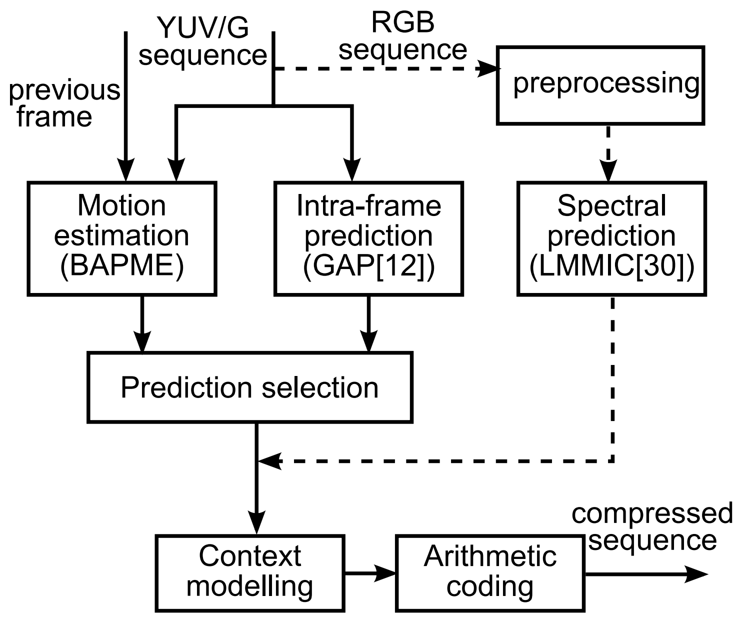


Figure 7: The complete lossless video compression system BAPME-VC.

colour videos, a spectral prediction technique is also included. Therefore, we combine the proposed motion estimation scheme BAPME, the intra-frame prediction scheme GAP [12] as well as the spectral prediction from our previous work [31] in the proposed video compression system.

4.1. System Overview

Fig. 7 illustrates the schematics of BAPME-VC. It follows the compression paradigm which proceeds as prediction, context modelling and arithmetic coding. The prediction part consists of motion estimation (BAPME), intra-frame prediction (GAP [12]) and spectral prediction (LMMIC [31]). Since GAP is used in the initial MV prediction, it is easy to reuse it. While all of them are available for any input video sequence, some are more efficient than others under different circumstances. Here we are mainly concerned about the most common types of sequences, YUV and RGB sequences.

For YUV sequences, since they are already colour decorrelated by the RGB-to-YUV transform, we apply the motion estimation and intra-frame prediction on each component. A prediction selection scheme is designed to select a better prediction based on the past performance of both predictors in the local area. We will give the formulation for this selection later in this section. The solid lines in Fig. 7 show the route that YUV video compression should go.

For RGB sequences, there exists a large amount of spectral redundancy. Therefore, we apply motion estimation and intra-frame prediction on the green (G) band, and process the red (R) and blue (B) bands in a similar manner as [31]. Band R and B are preprocessed to obtain the band difference and then are processed by the spectral prediction, which is a band-shifting based on a median predictor [31]. In our experiments, spectral prediction works better than using BAPME and GAP on band R and B, possibly due to the spectral correlation is more significant in RGB sequences and is easier to remove. The band G of the RGB sequence follows the solid lines in Fig. 7, while the R and B band follow the dash lines. Context modelling [31] and arithmetic coding are shared by both routes.

The prediction selection takes the prediction errors from BAPME and GAP, and adaptively decides which prediction error to output. It first calculates the sum of errors of each predictor in a local neighbourhood:

$$E = e_W + e_N + e_{NW} + e_{NE} + e_{WW} + e_{NN} + e_{NNE} \tag{10}$$

where e_W is the predicted error of a certain predictor on pixel W, and so on for others. The predictor which gives a smaller E is regarded as good predictor in the local neighbourhood and hence the corresponding prediction error is sent to the context modelling module. Our experiment result shows that this selection gives a right ratio of about 65% on average.

Similar to image compression, context modelling is also used in video compression to further exploit the higher-order redundancy. We use a similar model as in CALIC [12]. But different parameters are used to account for the different characteristics of the residual resulting from BAPME, GAP and spectral prediction. The contexts need be to constructed using the neighbouring pixels W, N, NW, NE, WW, NN and NNE. For intra-frame prediction GAP, the original pixels in current frame are used. For motion estimation BAPME, the differences between the pixels in current frame and the pixels in the motion compensated target window in previous frame are used instead. For spectral prediction, the band differences are used as the same way in [31]. This modelling scheme, although not yet fine tuned, improves the compression performance, as will be demonstrated below.

4.2. Experimental Performance

We present the experimental performance of BAPME-VC for YUV and RGB sequences respectively in this subsection.

YUV Sequences. For YUV sequences, we use two sets of commonly used test video sequences in CIF and QCIF format. We compare our compression scheme BAPME-VC with JPEG-LS [32], CALIC [12], ADAP [4], LI [16] and H.264/MPEG-4 AVC lossless mode [13]. JPEG-LS is the lossless image compression standard. CALIC is the benchmark algorithm for lossless image compression. ADAP is a block-based lossless video compression scheme using integer wavelet transform and motion estimation. LI is a pixel-based scheme that integrates motion estimation and various types of integer wavelet transform. LI [16] is claimed to achieve one of the best compression ratios in literature. H.264/MPEG-4 AVC is the latest standard for video coding using block-oriented motion estimation. The results here are from its lossless mode. The results of JPEG-LS, CALIC, ADAP and LI in Table 11 and Table 12 are extracted from [4] and [16]. However, due to the unavailability the software of ADAP and LI, it is impossible for us to carry out a thorough comparison on all sequences, and hence the blank in some parts of the tables. For H.264, we run the widely available software implementation x264 (version 0.83.1391) [33] in our experiments. We specify the main configuration as: 1 reference frame, 3 B-frames between I- and P-frames, 32 pixels search range, enabled CABAC (Context-Adaptive Binary Arithmetic Coding) and variable partitions. Based on this configuration, we run x264 with both integer-pel (IP) and sub-pel (SP) motion estimation. The sub-pel option uses SAD for mode decision and one quarter-pel iteration. In the following comparison, BAPME-VC only operates on integer-pel precision. And different from LI [16] which optimizes the search ranges for each video sequence, we simply fix our search range to ± 32 in the following comparison.

Table 11 presents the compression ratios of the selected schemes on the CIF YUV sequences. It shows that BAPME-VC outperforms all the integer-pel schemes in comparison on every sequences. It also works better than H.264 sub-pel on sequence "Paris" and "Hall", while is slightly worse than H.264 sub-pel on sequence "foreman". On average, BAPME-VC achieves 22% better bit rate over JPEG-LS, 21% better than CALIC, 5% better than ADAP, over 7% better than H.264 integer-pel and 4.8% better than H.264 sub-pel. It also outperforms LI by 0.08bpp on "football", even though it is a

Table 11: Compression rates comparison of selected schemes on CIF YUV sequences, in bbp.

sequence	Foreman	Paris	Hall	Average
JPEG-LS	4.99	5.89	4.90	5.26
CALIC	4.83	5.89	4.81	5.18
ADAP	4.70	3.57	4.66	4.31
LI	4.478			
H.264 (IP)	4.65	3.51	5.12	4.43
H.264 (SP)	**4.32**	3.45	5.12	4.30
BAPME-VC	4.40	**3.35**	**4.51**	**4.09**
gain over JPEG-LS (%)	11.78	43.09	7.95	22.28
gain over CALIC (%)	8.97	43.01	6.23	21.02
gain over ADAP (%)	6.44	5.97	3.15	5.12
gain over H.264 (IP) (%)	5.45	4.42	11.90	7.67
gain over H.264 (SP) (%)	-1.84	2.68	11.83	4.80

sequence with high motion activity and our search range is smaller than the one used in LI ((32×2+1) vs. 80).

The experimental performance on QCIF YUV sequences are shown in Table 12. BAPME-VC outperforms others on most of the sequences. It obtains 36% bit rate reduction over JPEG-LS, 35% over CALIC, 7% over ADAP and almost 4% over LI. Our scheme has a larger compression gain over JPEG-LS and CALIC on QCIF than CIF sequences, probably due to the fact that there is more temporal redundancy in smaller frame size sequences. BAPME-VC also works better than H.264 integer-pel, and achieves comparable bit rates with the H.264 sub-pel. Combined with the results from Table 11, BAPME has shown great potential to excel since its integer-pel version is already comparable with the H.264 sub-pel version. As presented in Section 3, the most simple version of sub-pel can already give improvement to BAPME. We expect even better performance when our scheme is equipped with a finer sub-pel BAPME, which will be part of our future work.

RGB Sequences. The compression ratios of selected schemes on RGB sequences are shown in Table 13. The results of ADAP and LI are absent due to the unavailability of the software, and x264 does not support RGB input. But we can still see the compression gain of BAPME-VC over JPEG-LS

Table 12: Compression rates comparison of selected schemes on QCIF YUV sequences, in bpp.

sequence	Foreman	News	Salesman	Silent	Mother	Akiyo	Carphone	Claire	Coastguard	Average
JPEG-LS	5.87	5.06	5.86	4.33	5.25	4.33	4.98	3.59	6.02	5.03
CALIC	5.68	5.11	5.78	4.18	5.18	4.31	4.93	3.67	5.99	4.98
ADAP	4.78	2.33	3.41	3.49	3.81	1.55	4.28	2.53	4.87	3.45
LI	4.64	2.20	3.28	2.84	3.73					
H.264 (full-pel)	4.96	2.20	3.48	3.13	3.42	1.38	4.65	2.60	4.68	3.39
H.264 (sub-pel)	**4.48**	**2.14**	3.44	3.03	3.28	**1.34**	4.35	**2.51**	**4.42**	3.22
BAPME-VC	4.61	2.15	**3.29**	**2.82**	**3.22**	1.49	**4.27**	2.54	4.49	**3.21**
gain over JPEG-LS (%)	21.48	57.50	43.86	34.81	38.78	65.67	14.10	29.22	25.39	36.23
gain over CALIC (%)	18.85	57.87	43.04	32.52	37.86	65.51	13.38	30.69	25.03	35.57
gain over ADAP (%)	3.67	7.69	3.41	19.07	15.65	3.91	0.22	-0.29	7.73	6.99
gain over Li (%)	0.66	2.37	-0.30	0.53	13.78					3.88
gain over H.264 (full-pel) (%)	7.13	2.27	5.31	9.68	5.91	-7.98	8.18	2.15	4.10	5.29
gain over H.264 (sub-pel) (%)	-2.90	-0.34	4.40	6.76	1.80	-11.29	1.70	-1.05	-1.68	0.35

Table 13: Compression rates comparison of selected schemes on RGB sequences, in bpp.

sequence	JPEG -LS	CALIC	BAPME -VC	gain over JPEG-LS(%)	gain over CALIC(%)
Football	14.41	13.79	**11.04**	23.34	19.92
Mobile	14.08	13.62	**11.36**	20.04	17.37
Susie	11.26	10.77	**9.60**	14.76	10.86
Missa	11.27	10.94	**8.36**	25.78	23.53
Claire	7.41	7.07	**6.02**	18.83	14.88
Average	11.79	11.36	**9.17**	20.55	17.31

and CALIC, which is about 20% and 17% on average. As expected, more bits reduction is achieved on sequences with more motion activity, such as “football” and “mobile”.

5. Conclusion

In this paper, we considered the possibility of borrowing strength from both pixel-based algorithms and block-matching algorithms. We proposed a novel Backward Adaptive pixel-based fast Predictive Motion Estimation (BAPME) scheme. It operates on a pixel-by-pixel basis to provide highly accurate motion estimation without transmitting any side information, while taking advantage of fast search to maintain low complexity. In contrast with block-matching algorithms where full search generally achieves better motion estimation accuracy than fast search, BAPME outperforms the *pixel-based* full search in our experiments. BAPME also obtains better overall zero-order entropy than *block-based* full search when taking into account the side information of block-based schemes. In our complexity analysis, BAPME is faster than block-based full search. Moreover, the half-pel version of BAPME shows promising results. We also presented the complete lossless video compression system BAPME-VC, which integrates integer-pel BAPME and lossless image compression techniques for intra-frame prediction. BAPME-VC outperforms other state-of-the-art integer-pel schemes on three sets of test sequences and achieves comparable bit rates with H.264 lossless mode sub-pixel version. Overall, we believe that pixel-based motion estimation has shown its potential in contributing to modern video compression systems.

6. Acknowledgment

The authors are grateful for the support from ESA and EPSRC under grant EP/D011639/1.

References

[1] E. Carotti, J. Martin, A. Meo, Backward-adaptive lossless compression of video sequences, in: Proc. IEEE Int. Conf. on Audio, Speech, Signal Processing, 2002, pp. 3417–3420.

[2] R. Oami, M. Ohta, Efficient lossless video coding compatible with MPEG-2, in: Proc. IEEE Int. Conf. on Communications, Vol. 2, 1998, pp. 901–905.

[3] Y. Gong, S. Pullalarevu, S. Sheikh, A wavelet-based lossless video coding scheme, in: Proc. Int. Conf. on Signal Processing, 2004, pp. 1123–1126.

[4] S.-G. Park, E. J. Delp, H. Yu, Adaptive lossless video compression using an integer wavelet transform, in: Proce. IEEE Int. Conf. on Image Processing, 2004, pp. 2251–2254.

[5] T. Qiu, X. Wu, Z. Xiao, Scalable lossy to lossless video coding via adaptive 3d wavelet transform and context modelling, in: Proce. IEEE Int. Conf. on Image Processing, Vol. 2, 2000, pp. 855–858.

[6] H.-W. Park, H.-S. Kim, Motion estimation using low-band-shift method for wavelet-based moving-picture coding, IEEE Trans. Image Process. 9 (4) (2000) 577–587.

[7] N. Memon, K. Sayood, Lossless compression of video sequences, IEEE Trans. Commun. 44 (10) (1996) 1340–1345.

[8] D. Brunello, G. Calvagno, G. Mian, R. Rinaldo, Lossless compression of video using temporal information, IEEE Trans. Image Process. 12 (2) (2003) 132–139.

[9] M.-F. Zhang, J. Hu, L.-M. Zhang, Lossless video compression using combination of temporal and spatial prediction, in: Proc. of Int. Conf. on Neural Networks and Signal Processing, Vol. 2, 2003, pp. 1193–1196.

[10] I. Matsuda, T. Shiodera, S. Itoh, Lossless video coding using variable block-size MC and 3D prediction optimized for each frame, in: Proc. European Signal Processing Conf., 2004, pp. 1967–1970.

[11] E. Carotti, J. Martin, Motion-compensated lossless video coding in the CALIC framework, in: Proc. IEEE Symp. on Signal Processing and Information Technology, 2005, pp. 600–605.

[12] X. Wu, N. Memon, Context-based, adaptive, lossless image coding, IEEE Trans. Commun. 45 (4) (1997) 437–444.

[13] I. E. G. Richardson, H.264 and MPEG-4 Video Compression – Video Coding for Next-generation Multimedia, Wiley, 2003.

[14] K. Yang, A. Faryar, A context-based predictive coder for lossless and near-lossless compression of video, in: Proce. IEEE Int. Conf. on Image Processing, Vol. 1, 2000, pp. 144–147.

[15] E. Carotti, J. Martin, A. Meo, Low-complexity lossless video coding via adaptive spatio-temporal prediction, in: Proce. IEEE Int. Conf. on Image Processing, Vol. 2, 2003, pp. 197–200.

[16] Y. Li, K. Sayood, Lossless video sequence compression using adaptive prediction, IEEE Trans. Image Process. 16 (4) (2007) 997–1007.

[17] Z. Chen, J. Xu, Y. He, J. Zheng, Fast integer-pel and fractional-pel motion estimation for H.264/AVC, Journal of Visual Communication and Image Representation 17 (2) (2006) 264–290.

[18] W. I. Choi, B. Jeon, J. Jeong, Fast motion estimation with modified diamond search for variable motion block sizes, in: Proce. IEEE Int. Conf. on Image Processing, Vol. 2, 2003, pp. 371–374.

[19] A. Tourapis, O. Au, M. Liou, Predictive motion vector field adaptive search technique (PMVFAST) enhancing block based motion estimation, in: Proc. Visual Commun. and Image Processing, 2001.

[20] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, T. Ishiguro, Motion compensated interframe coding for video conferencing, in: Proc. National Telecommunication Conference, 1981, pp. G5.3.1–G5.3.5.

[21] J. Tham, S. Ranganath, M. Ranganath, A. Kassim, A novel unrestricted center-biased diamond search algorithm for block motion estimation, IEEE Trans. Circuits Syst. Video Technol. 8 (8) (1998) 369–377.

[22] S. Zhu, K. Ma, A new diamond search algorithm for fast block-matching motion estimation, IEEE Trans. Image Process. 9 (2) (2000) 287–290.

[23] C. Zhu, X. Lin, L. Chau, Hexagon-based search pattern for fast block motion estimation, IEEE Trans. Circuits Syst. Video Technol. 12 (5) (2002) 349–355.

[24] X. Chen, N. Canagarajah, J. L. Nunez-Yanez, Backward adaptive pixel-based fast predictive motion estimation, IEEE Signal Process. Lett. 16 (5) (2009) 370–373.

[25] C.-H. Cheung, L.-M. Po, A novel cross-diamond search algorithm for fast block motion estimation, IEEE Trans. Circuits Syst. Video Technol. 12 (12) (2002) 1168–1177.

[26] C.-H. Cheung, L.-M. Po, Novel cross-diamond-hexagonal search algorithms for fast block motion estimation, IEEE Trans. Multimedia 7 (1) (2005) 16–22.

[27] M. Ueda, TE1: Refinement motion compensation using decoder-side motion estimation, Tech. rep., JCT-VC (Jul. 2010).

[28] C. Zhu, X. Lin, L. Chau, L.-M. Po, Enhanced hexagonal search for fast block motion estimation, IEEE Trans. Circuits Syst. Video Technol. 14 (10) (2004) 1210–1214.

[29] Society of Motion Picture and Television Engineers (SMPTE) 421M, VC-1 Compression Video Bitstream Format and Decoding Process, SMPTE, 2005.

[30] W. Lin, D. Baylon, K. Panusopone, M.-T. Sun, Fast sub-pixel motion estimation and mode decision for H.264, in: Proc. IEEE Int. Symp. on Circuits and Systems, 2008, pp. 3482–3485.

[31] X. Chen, N. Canagarajah, J. Nunez-Yanez, Lossless multi-mode inter-band image compression and its hardware architecture, in: Proc. Conf. on Design and Architecture for Signal and Image Processing, 2008, pp. 208–215.

[32] M. J. Weinberger, G. Seroussi, G. Sapiro, LOCO-I: A low complexity, context-based, lossless image compression algorithm, in: Proc. Data Compression Conf., 1996, pp. 140–149.

[33] V. Organization, x264, `http://www.videolan.org/developers/x264.html`.