Predictive and adaptive rood pattern with large motion search for H.264 video coding

Citation for published version (APA):

DOI:
10.1117/1.3227902

Document status and date:
Published: 01/01/2009

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication
Predictive and adaptive rood pattern with large motion search for H.264 video coding

Hong Yin Lim
Ashraf A. Kassim
National University of Singapore
Department of Electrical and Computer Engineering
4 Engineering Drive 3
Singapore 117576
E-mail: ashraf@nus.edu.sg

Peter H. N. de With
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven
The Netherlands

Abstract. We propose a fast motion estimation (FME) algorithm that performs comparably to the full search while having improved performance when compared to the best FME methods that are recommended for the JVT/H.264 standard despite having a much lower computational complexity than these methods. Our algorithm, called the predictive and adaptive rood pattern with large motion search, incorporates motion vector prediction using spatial and temporal correlation, an adaptive search pattern, multiple refinement search paths, and an adaptive moving search window scheme that is specifically designed for searching large and complex motion.

1 Introduction

In current video coding standards, block-matching motion estimation and compensation are used to remove the interframe redundancy in the video. In the H.264 standard, the rate-distortion (RD) Optimization is used in the block-matching criterion, where the optimal motion vector (MV) is determined from the minimization of a cost function involving the sum of absolute difference and the number of bits needed to code the MV. The most common method used for the block-matching motion estimation is the full search (FS), where the search is conducted considering all possible displaced locations within the search area in the reference frame. This ensures the best MV is found, but unfortunately it is too time consuming. Although the best MV is important for video coding, the FS is not appropriate for applications such as video denoising and video deinterlacing. The computational load using FS is made worse when concepts such as variable block sizes (seven different block sizes in H.264) and multiframe motion estimation (maximum of 32 reference frames) is used.

In order to reduce the high computational complexity associated with FS, fast motion estimation (FME) algorithms were developed that evaluate a fewer number of search points. Some well-known FME algorithms include the diamond search, adaptive rood pattern search (ARPS-2), enhanced predictive zonal search (EPZS), fast adaptive motion estimation (FAME), and unsymmetrically multigrid hexagonal search (UMHexagonS). The more recent FME algorithms, such as EPZS, FAME and UMHexagonS, use motion vector prediction to estimate the best MV.

In most FME algorithms, the quality is severely affected (a loss of about 1–3 dB) when the video frame contains large and complex motions (see Fig. 16). The large and complex motion is due to the background/scene changes (camera panning motion) and fast movement of the object. High interframe motion may be also due to the low video coding frame rate. These videos will be termed as high-motion sequences; while videos that have almost static motion will be termed low-motion sequences. In this paper, we propose a FME algorithm with low computational complexity and able to maintain good video quality, especially for high-motion sequences. Our proposed algorithm uses novel MV prediction techniques and a movable search window for searching large and complex motions.

The rest of the paper is organized as follows. Section 2 provides an overview of some predictive FME algorithms. In Secs. 3 and 4, we introduce our predictive and adaptive rood pattern with large motion search (PARPLMS) algorithm and compare its performance with other FME algorithms in Sec. 5. The final section concludes this paper.

2 Recent FME Algorithms

In predictive FME algorithms, such as EPZS, FAME, and UMHexagonS, a set of candidate predictors are initially evaluated in order to estimate the best MV. These predictors include the MVs of spatially adjacent blocks, shown in
Fig. 1, and the (0,0) MV (i.e., the static block). EPZS and FAME also uses the MVs of temporally adjacent blocks (Fig. 2) for its candidate predictors. Because of the multi-reference frame and variable block size methods introduced in H.264,1 additional candidate predictors such as the neighboring reference frame predictor and the up-layer predictor (see Sec. 3.2) are also used in EPZS and UMHexagonS.

In these predictive FME algorithms (Fig. 3), the best predictor is identified from the candidate predictors as the one with the minimum cost function. This cost function is then compared to a threshold (early-termination criteria) and, if it is less than the threshold, then the best predictor is determined as the best MV for the block and the MV search is terminated. Otherwise, a MV refinement search with the resulting best predictor forming the search center is carried out. The pattern used in the search is determined by the algorithm based on the motion activity of the block. Fixed patterns such as the hexagon search pattern8 (UMHexagonS) or large diamond pattern3 is used for high-motion activity (coarse refinement search) while the small diamond pattern (Fig. 4)3 is used for low-motion activity (fine refinement search). More details about the predictive FME algorithms are found in Refs. 5–8.

3 Overview of PARPLMS

Figure 5 provides an overview of our proposed predictive and adaptive rood pattern with large motion search (PARPLMS) algorithm. PARPLMS uses MV prediction, where the best predictors from each category consisting of the spatial, up-layer, temporal, and neighboring reference predictors, forms search centers for different refinement search paths. Using adaptive refinement search based on the local motion activity of neighboring blocks, the best MV obtained from each refinement search path is compared and the overall best MV is then determined. A new large motion search strategy, called the extended rood search, is used for searching large and complex motions. PARPLMS uses an adaptive moving search window method, where several other candidates are also considered as the center of the search window.

As shown in Refs. 5–8, the MVs of spatially adjacent blocks and temporally adjacent blocks are highly correlated with the MV of the current block. In a homogeneous region, the optimal MV is usually found near or at the same location as one of the MV predictors. Therefore, the MV predictors are used to estimate a search center where further refinement search can be done. In our algorithm, the candidate MV predictors used are divided into the following four sets.
3.1 Set A: Spatial Predictors

Four possible candidate spatial predictors are the MVs from the spatially adjacent blocks (see Fig. 1): left, top, top-right, and the stationary block \([0,0]\) MV. The MVs of the left, top, and top-right blocks are used because they are sufficient as predictors for the current MV. In certain locations, when some of the predictors may not be available (i.e., in last column, MV of the top-right block is not available), the MV of the top-left block is used as a substitute.

3.2 Set B: Uplayer Block Predictor(s)

In the H.264 reference software, the MVs of different block sizes are estimated in a top-down approach, where the MVs of bigger block sizes (up-layer) are estimated first. Because of a strong correlation between the different blocks, the MVs of higher-level blocks can be used as candidate predictors for the MVs of the lower-level blocks. As shown in Fig. 6, there is a significant increase (as much as 16\%) in the number of better MVs found (with smaller cost function) when candidates from subblocks are included.

PARPLMS also uses all MVs of the uplayer block sizes as candidate predictors. Therefore for a \(4 \times 4\) block, the MVs of the \(4 \times 8\), \(8 \times 4\), \(8 \times 8\), \(8 \times 16\), \(16 \times 8\), and \(16 \times 16\) block sizes are also used as its predictors.

3.3 Set C: Temporal Predictors

The set of temporal predictors consists of the MV of the colocated block (block at the same location from the previous encoded frame) and the MVs of four subsequent adjacent blocks: right, bottom-left, bottom, and bottom-right in the previous encoded frame. Our temporal predictors are located after the colocated block shaded portion of Fig. 2, which avoids the need for extra memory to store the temporal predictors because we use the same memory to store the temporal predictors associated with the previous frame and spatial predictors for the current frame.

As the temporal distance between the reference frame and the current frame increases, the temporal predictors lose its correlation with the current MV. Therefore, for the second reference frame onward (\(\text{ref} > 0\)), the predictors of set C are not used. This also leads to a reduction in the computational complexity, especially when evaluating further reference frames that are less likely to be correlated with the current frame.

3.4 Set D: Neighboring Reference Frame Predictor

Because the motion prediction in H.264 is performed based on multiple reference frames, the best MV obtained from the previous encoded frame can be used as a predictor for the current reference frame. The basis of Ref. 8, the linear relation between the MV found from one reference frame to the next reference frame can be determined as follows:

\[
\text{MV}_{\text{current}} = \text{MV}_{\text{previous}} + \alpha \cdot \text{MV}_{\text{subblock}}
\]

Fig. 7 Adaptive rood pattern.
\[ MV_t = \frac{MV_c \times D_t}{D_c}, \]  

(1)

where \( MV_t \) is the predicted MV for the current reference frame, \( MV_c \) is the best MV obtained from the previous reference frame, \( D_t \) is the temporal distance between the current reference frame and the frame being encoded, and \( D_c \) is the temporal distance between the previous reference frame and the frame being encoded. This relation holds if we assume that the picture is undergoing a linear motion, such as a smooth pan or tilt motion, which is normally associated with videos.

4 Search Methods used in PARPLMS

For the coarse refinement search, the adaptive rood pattern (Fig. 7) of the ARPS-2 algorithm\(^4\) is used with the center positioned at the best predictor. The novelty of the adaptive rood pattern is that its size is adaptively determined based on the motion activity of adjacent blocks. This ensures that the coarse refinement search is confined within a suitable range from the best MV predictor, and hence, only the relevant MVs are evaluated. At each search iteration \( i \), the rood arm size is determined as follows:

\[ R_x = 1/2[i \times [\text{MAX}(MV_x) - \text{MIN}(MV_x)]]; \quad i = 1, 2, \ldots \]

Fig. 8 PARPLMS's shrinking adaptive rood pattern search method.

Fig. 9 Extended rood pattern, where SR is the maximum search range.
pattern search

is supported by the spatially adjacent components, respectively, of the selected MVs in the region of RX and the initial root pattern is set to 4. At the beginning of the search, the root pattern size is adaptively determined from Eq. (2) (with \(i = 1\)) and the center of the root pattern is placed on the best predictor obtained from the prediction. The points of the root pattern are then evaluated, and the best MV is determined. The root pattern is reduced in size (i.e., shrunk) by half at each search iteration (\(i = 1, 2, \ldots\)) and is centered on the best MV. The process is repeated until the root arm size reduces to one; at which point it is equivalent to the small diamond search (SDS). PARPLMS’s shrinking adaptive root pattern search refinement search strategy is illustrated in Fig. 8.

Figure 9 compares PARPLMS’s shrinking adaptive root pattern search [Fig. 9(a)] with a nonshrinking search method [Fig. 9(a)], where the root arm size is kept constant at each iteration. The nonshrinking root pattern search method is clearly unable to cover the horizontal area between 0 and 3, making it ineffective for refining large motion search. Furthermore, the search is trapped easily in a local minimum. In contrast, PARPLMS’s shrinking adaptive root pattern search is able to cover a wider area (including the horizontal area between 0 and 3), as seen in Fig. 9(b). Moreover, there is no increase in computational complexity when estimating low-motion video sequences, because PARPLMS’s shrinking adaptive root pattern search is usually bypassed (because \(R_x = 4\) and \(R_y = 2\)) and the small diamond search is used. Thus, the performance of PARPLMS for low-motion video sequences will be similar to other predictive FME algorithms.

4.1 Multiple Refinement Search Paths

Selecting the best predictor from a combined set of all predictors (spatial, uplayer, temporal, neighboring reference) may not result in the optimal MV. For example, the optimal MV may lie near the spatially correlated predictors (sets A and B), but the best predictor found during prediction could very well be from the temporally correlated predictors (sets C and D). For this reason, we use multiple search paths based on the different set of predictors in our algorithm. In the spatially correlated search, the best predictor is first obtained from set A. PARPLMS’s refinement search using the shrinking adaptive root pattern is then performed on the best predictor and the best MVs are stored along with the cost function. Next, the SDS is performed on the best predictor from set B and, similarly, the best MV is stored along with the cost function.

In the temporally correlated search for the first reference frame (ref = 0), the best predictor is obtained from set C, which will form the search center for PARPLMS’s refinement search. Because the temporal predictors might be different from the spatial predictors, the region of support to determine the root pattern size is formed using the temporal predictors instead. The MVs of the colocated, right, and bottom block (see Fig. 2) are used to calculate the root pattern size. Similarly, if any of the predictors used in calculating the root pattern size is unavailable, either the bottom-left and/or bottom-right MV can be used. For the

\[
R_y = \frac{1}{2} [\text{MAX}(MV_y) - \text{MIN}(MV_y)],
\]

where \(MV_x\) and \(MV_y\) are the horizontal and vertical components, respectively, of the selected MVs in the region of support formed by the spatially adjacent predictors (Fig. 1).

In PARPLMS, the MVs used are those of the left, top, and top-right blocks, with the top-left MV used as a substitute in locations where some predictors are not available, and the initial root pattern is set to \(R_x = R_y = 4\) at the first row or if the region of support has only one element. At the beginning of the search, the root pattern size is adaptively determined from Eq. (2) (with \(i = 1\)) and the center of the root pattern is placed on the best predictor obtained from the prediction. The points of the root pattern are then evaluated, and the best MV is determined. The root pattern is reduced in size (i.e., shrunk) by half at each search iteration (\(i = 1, 2, \ldots\)) and is centered on the best MV. The process is repeated until the root arm size reduces to one; at which point it is equivalent to the small diamond search (SDS). PARPLMS’s shrinking adaptive root pattern search refinement search strategy is illustrated in Fig. 8.

Figure 9 compares PARPLMS’s shrinking adaptive root pattern search [Fig. 9(a)] with a nonshrinking search method [Fig. 9(a)], where the root arm size is kept constant at each iteration. The nonshrinking root pattern search method is clearly unable to cover the horizontal area between 0 and 3, making it ineffective for refining large motion search. Furthermore, the search is trapped easily in a local minimum. In contrast, PARPLMS’s shrinking adaptive root pattern search is able to cover a wider area (including the horizontal area between 0 and 3), as seen in Fig. 9(b). Moreover, there is no increase in computational complexity when estimating low-motion video sequences, because PARPLMS’s shrinking adaptive root pattern search is usually bypassed (because \(R_x = 4\) and \(R_y = 2\)) and the small diamond search is used. Thus, the performance of PARPLMS for low-motion video sequences will be similar to other predictive FME algorithms.

4.1 Multiple Refinement Search Paths

Selecting the best predictor from a combined set of all predictors (spatial, uplayer, temporal, neighboring reference) may not result in the optimal MV. For example, the optimal MV may lie near the spatially correlated predictors (sets A and B), but the best predictor found during prediction could very well be from the temporally correlated predictors (sets C and D). For this reason, we use multiple search paths based on the different set of predictors in our algorithm. In the spatially correlated search, the best predictor is first obtained from set A. PARPLMS’s refinement search using the shrinking adaptive root pattern is then performed on the best predictor and the best MVs are stored along with the cost function. Next, the SDS is performed on the best predictor from set B and, similarly, the best MV is stored along with the cost function.

In the temporally correlated search for the first reference frame (ref = 0), the best predictor is obtained from set C, which will form the search center for PARPLMS’s refinement search. Because the temporal predictors might be different from the spatial predictors, the region of support to determine the root pattern size is formed using the temporal predictors instead. The MVs of the colocated, right, and bottom block (see Fig. 2) are used to calculate the root pattern size. Similarly, if any of the predictors used in calculating the root pattern size is unavailable, either the bottom-left and/or bottom-right MV can be used. For the
case when the temporal region of support has only 1 ele-
ment (for the last row and column of blocks, only the co-
lated MV is available), $R_x = R_y = 4$. In the temporally corre-
lated search for the second reference frame onward (ref $> 0$), only a single search path is conduction because the pre-
dictors of set C are not used. In this search, the SDS is
performed on the predictor from set D.

Because the best MVs are usually found from the near-
est reference frame (ref $= 0$), restricting the use of the re-
finement search in further reference frames would reduce
the computational complexity of the search process. Be-
cause the optimal MV usually lies near the set A predictors,
we restrict the SDS performed on the best predictor of sets
B and D for the second reference frame onward (ref $> 0$).
The SDS is only performed for evaluation of points around
sets B and D when the minimum cost function, $J_{\text{min}}$ is less
than a threshold, $T_{\text{ref}}$.

$$T_{\text{ref}} = \beta \times J_{\text{min}} \; \text{(of best reference thus),} \tag{3}$$

where $\beta$ is a fixed parameter, which ensures that the refine-
ment search is performed only if $J_{\text{min}}$ of the current refer-
ence is not considerably higher than the best $J_{\text{min}}$ found
from previous reference frames. In our experiments, we set $\beta$
$= 1.2$.

At the end of all the different refinement search paths (due to sets A–D), the minimum cost function from each
search path is compared and the best cost function and the corresponding MV is determined.

### 4.2 Large Motion Search—Extended Rood Search Method

Because not every block may contain large and complex
motion, it makes sense that a different search method be
used only for high-motion activity blocks. To determine the
motion activity, the minimum cost functions ($J_{\text{min}}$) of the
spatially adjacent blocks are used; similar to the adaptive
early-termination criteria introduced in Ref. 6. If the current
$J_{\text{min}}$ found from the refinement search around the set of
predictors is significantly higher than the $J_{\text{min}}$ of the adja-
cent blocks, then it would imply that the optimal MV is
significantly different from the best MV obtained from the
refinement search and hence is a good indication of high-
motion activity. From our experiments, we found that the
search for large motion should take place when the best
cost function found from the multiple refinement search
paths around the set of predictors is larger than a threshold
that is twice the lowest of the minimum cost functions of
the three spatially adjacent blocks: left, top, and top right.

Table 1 shows that high-motion video sequences (i.e.,
Stefan and Bus) generally have a high percentage ($>70\%)$
of large MVs. To effectively locate these large magnitude
MVIs for such high-motion video sequences, we use a large
motion search strategy, which first identifies the main mo-
tion path and then refines the search to specifically locate
small deviations from the main motion path.

Our large motion search strategy uses a new search pat-
tern called the extended rood pattern shown in Fig. 10.

### Table 1 Large MVs in the bus, Foreman (Fore), and Stephan (Stef) test sequences at 10 fps (Bus10, Fore10, Stef10) and 15 fps, (Bus15, Fore15, Stef15).

<table>
<thead>
<tr>
<th>Magnitude of MV</th>
<th>Bus15</th>
<th>Bus10</th>
<th>Stef15</th>
<th>Stef10</th>
<th>Fore15</th>
<th>Fore10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>\text{Best MV}</td>
<td>&gt;4$</td>
<td>79.6</td>
<td>82.9</td>
<td>70.2</td>
<td>74.4</td>
</tr>
</tbody>
</table>

### Table 2 Low-motion blocks identified by the new criterion (% LM), which are also identified by the FS method as low-motion blocks (% Correct).

<table>
<thead>
<tr>
<th>Sequence</th>
<th>% LM</th>
<th>% Correct</th>
<th>Sequence</th>
<th>% LM</th>
<th>% Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hall</td>
<td>7.5</td>
<td>84.3</td>
<td>Bus</td>
<td>15</td>
<td>4.2</td>
</tr>
<tr>
<td>Hall</td>
<td>15</td>
<td>87.1</td>
<td>Bus</td>
<td>30</td>
<td>8.1</td>
</tr>
<tr>
<td>Mother</td>
<td>7.5</td>
<td>78.6</td>
<td>Football</td>
<td>15</td>
<td>12.7</td>
</tr>
<tr>
<td>Mother</td>
<td>15</td>
<td>83.8</td>
<td>Football</td>
<td>30</td>
<td>16.6</td>
</tr>
<tr>
<td>Container</td>
<td>7.5</td>
<td>80.8</td>
<td>Stefan</td>
<td>15</td>
<td>9.7</td>
</tr>
<tr>
<td>Container</td>
<td>15</td>
<td>82.6</td>
<td>Stefan</td>
<td>30</td>
<td>18.2</td>
</tr>
</tbody>
</table>
where the distance from the search window center (square dot) to each search point (circular dot) on the horizontal and vertical axes is a multiple of 4 (i.e., the distance between adjacent search points is 4). For example, a search range (SR) of \( \pm 32 \) would result in \( 2 \times [32 - (-32)] \times (1/4) = 32 \) search points. Although a distance of more than 4 between adjacent search points would not be useful for searching low-motion sequences, reducing it to 2 would double the number of search points. In the initial phase for the large and complex motion search, the extended rood pattern is used in order to locate the general direction and magnitude of the motion. After this, the large motion search is further refined by searching for small deviations from the main motion. This is done by using PARPLMS’s shrinking adaptive rood pattern starting with \( R_x = R_y = 2 \).

To further keep the computations to a minimum, the extended rood search is only used for block mode 1 (16 × 16 block sizes) and block mode 4 (8 × 8 block sizes) but not for the other block modes. This is because the 16 × 16 block can be used as an approximation for 16 × 8 and 8 × 16 block and 8 × 8 block as approximation for the remaining smaller (8 × 4, 4 × 8, and 4 × 4) block sizes.

4.3 Adaptive Moving Search Window

In the H.264 motion estimation process, the search window center is fixed at predicted MV [Fig. 11(a)]. However, due to the large and complex motion associated with the high-motion sequences, a better MV could be obtained either outside the maximum search range from the predicted MV or inside the maximum search range from the spatial median [Fig. 11(b)]. PARPLMS uses an adaptive moving search window scheme, where the search window is centered at the best predictor, which is the one with the minimum cost function from the candidate search window center predictors, as follows:

1. Predicted MV: median of the best MV of the left, top, and top-right blocks using the best block mode and best reference frame
2. Spatial median MV: median of the spatial predictors determined from the MVs of the left, top, and top-right blocks for the current block mode and reference frame
3. Temporal median MV: median of the temporal predictors determined from the MVs of the colocated, right, and bottom blocks

Figure 12 clearly shows that the average minimum cost function per block obtained using our adaptive moving search window scheme (scheme 2 in Fig. 12) results in a

![Figure 12: Average minimum cost function per block for different search window positions](image1)

![Figure 13: Average PSNR per frame versus bitrate for Football(CIF) @ 10fps](image2)

**Fig. 12** Average minimum cost function per block for different search window positions.

**Fig. 13** Average PSNR per frame versus bitrate for the Football sequence encoded at 10 fps.
Table 3 Performance of FME algorithms compared to FS for low- and medium-motion sequences.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>FR (fps)</th>
<th>Measures</th>
<th>Block motion estimation algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ΔPSNR (dB)</td>
<td>ARPS2</td>
</tr>
<tr>
<td>Hall 15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hall 10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mother 15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mother 10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Container 15</td>
<td>0.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Container 10</td>
<td>0.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coastguard 15</td>
<td>0.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coastguard 10</td>
<td>-0.06</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Lim, Kassim, and de With: Predictive and adaptive road pattern with large motion search…
lower minimum cost function (up to 7% less) compared to the fixed search window centered at the predicted MV (scheme 1 in Fig. 12).

4.4 Determination for Low-Motion Activity

When the best MVs obtained for low-motion sequences are examined, it was found that the majority of these lie within ±1 pixel from the origin. Also, temporally correlated predictors (i.e., sets C and D) are not useful for locating the best MV for low-motion sequences. Hence, to identify low-motion blocks, we use the following criterion:

If the best MV from the spatially correlated refinement search (sets A and B) and the spatially adjacent MVs (left, top, top right) is within ±1 pixel from the origin, the block is considered to have low-motion activity.

If the best MV from the spatially correlated refinement search (sets A and B) and the spatially adjacent MVs (left, top, top right) is within ±1 pixel from the origin, the block is considered to have low-motion activity.

Table 2 shows the results of implementing this criterion. For the low-motion sequences (Hall, Mother, and Container), over 97% of the blocks that are identified as low-motion blocks using the criterion are actually low-motion blocks as determined via full search. As mentioned above, the evaluation of the temporally correlated predictors can be skipped for low-motion blocks (i.e., sets C and D). In addition, when the block also satisfies the criterion for possible high-motion activity, the PARPLMS’s shrinking adaptive rood pattern starting with \( R_x = R_y = 4 \) is used instead of the extended rood search.

5 Experimental Results

The experiments were carried out using the JVT H.264 reference software\(^8\) for the following classes of video sequences:

1. Low-motion (QCIF)—Hall, Mother, Container
2. Medium-motion—Coastguard (QCIF), Foreman (CIF)
3. High-motion (CIF)—Bus, Football, Canoa, Stefan

The results (Tables 3 and 4 and Figs. 13–15) are presented for the encoding frame rates of 30, 15 and 10 frames per second (fps) and at different quantizer values: 28, 32, 36, 40.\(^{12}\) In our experiments, the H.264 main profile is used with the following encoding parameters: Hadamard transform, RD optimization, CABAC encoding, and no B slice. The experiments are conducted on a 3.6 GHz with 1 GB DDR-RAM Pentium IV computer.

Several measures are used to compare the performance of the FME algorithms:

1. \( \Delta \)PSNR—average increase in peak signal-to-noise ratio (PSNR) per frame, compared to FS. A negative value shows a loss of PSNR.
2. Total time—Percentage of the computational gain over FS in terms of the time used for the entire encoding process, which is computed as follows:
   \[
   \% \text{Total time} = \frac{T_{FS} - T_{FME}}{T_{FS}} \times 100 \%
   \]
   where \( T_{FS} \) is total encoding time for FS and \( T_{FME} \) is total encoding time for FME algorithms.
3. ME time—Percentage of the computational gain over FS in terms of the time used for the motion estimation process (for all block modes). Computation is the same as Eq. (4), except the total encoding time is replaced with total motion estimation time.
4. \( \Delta \)Bit rate—Percentage of savings in total number of bits needed to encode the sequence compared to FS. A positive value shows an increase in bits needed for the encoding.

We compare our proposed PARPLMS to existing FME algorithms based on the average PSNR gain, speed-up, MV computational time, and bit-rate reduction results obtained from the quantizer values recommended in Ref. 12 (i.e., 28, 32, 36, 40). The results shown for UMHexagonS and EPZS are obtained using the implementation of these algorithms in the JM software while the ARPS-2 algorithm is implemented based on Ref. 4. All comparisons are done with respect to the original FS algorithm and not the JM’s fast

<table>
<thead>
<tr>
<th>Table 3 (Continued.)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sequence</strong></td>
</tr>
<tr>
<td><strong>Foreman</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Foreman</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
**Table 4** Performance of FME algorithms compared to FS for high-motion sequences.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>FR (fps)</th>
<th>Measures</th>
<th>Block motion estimation algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>ARPS2</td>
</tr>
<tr>
<td>Bus 15</td>
<td></td>
<td>$\Delta$PSNR (dB)</td>
<td>-0.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total time (%)</td>
<td>92.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ME time (%)</td>
<td>96.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Delta$Bit rate (%)</td>
<td>30.68</td>
</tr>
<tr>
<td>Bus 10</td>
<td></td>
<td>$\Delta$PSNR (dB)</td>
<td>-0.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total time (%)</td>
<td>92.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ME time (%)</td>
<td>96.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Delta$Bit rate (%)</td>
<td>40.34</td>
</tr>
<tr>
<td>Football 15</td>
<td></td>
<td>$\Delta$PSNR (dB)</td>
<td>-0.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total time (%)</td>
<td>93.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ME time (%)</td>
<td>97.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Delta$Bit rate (%)</td>
<td>8.88</td>
</tr>
<tr>
<td>Football 10</td>
<td></td>
<td>$\Delta$PSNR (dB)</td>
<td>-0.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total time (%)</td>
<td>93.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ME time (%)</td>
<td>97.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Delta$Bit rate (%)</td>
<td>7.63</td>
</tr>
<tr>
<td>Canoa 15</td>
<td></td>
<td>$\Delta$PSNR (dB)</td>
<td>-0.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total time (%)</td>
<td>92.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ME time (%)</td>
<td>96.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Delta$Bit rate (%)</td>
<td>7.04</td>
</tr>
<tr>
<td>Canoa 10</td>
<td></td>
<td>$\Delta$PSNR (dB)</td>
<td>-0.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total time (%)</td>
<td>92.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ME time (%)</td>
<td>96.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Delta$Bit rate (%)</td>
<td>15.29</td>
</tr>
<tr>
<td>Stefan 30</td>
<td></td>
<td>$\Delta$PSNR (dB)</td>
<td>-0.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total time (%)</td>
<td>92.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ME time (%)</td>
<td>96.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Delta$Bit rate (%)</td>
<td>16.55</td>
</tr>
<tr>
<td>Stefan 15</td>
<td></td>
<td>$\Delta$PSNR (dB)</td>
<td>-0.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total time (%)</td>
<td>91.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ME time (%)</td>
<td>96.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Delta$Bit rate (%)</td>
<td>47.69</td>
</tr>
</tbody>
</table>
FS. The SR used for the motion estimation is ±16 and ±32 for the QCIF and CIF sequences, respectively. For the low-motion sequences, two reference frames are used because most of the best MVs are usually located within the first two frames. For the other sequences, five reference frames are used. FR represents the frame rate used in the encoding process.

Figures 13–15 show the RD plots obtained for different high-motion sequences at 10 fps. A lower frame rate was used to simulate large and complex motion. Clearly, PARPLMS outperforms UMHexagonS and EPZS, which are recommended for the JVT/H.264 standard, and even outperforms the best possible performance obtained using FS in Fig. 15. The performance of the ARPS-2 algorithm is also much worse than PARPLMS. Even though ARPS-2 has the highest savings in computational time it has the lowest/worst PSNR performance.

Figure 16 shows the PSNR variations for selected frames of the high-motion Stefan sequence encoded at 10 fps and at a fixed medium encoding bit rate of 500 kbps with rate control enabled. Frames 234–279 of the Stefan sequence contain large and complex motion due to huge panning motions, resulting in rapid changes to the background view. In addition, the object also has its own independent motion. For frames 240–273, PARPLMS performs better by up to 4 dB when compared to UMHexagonS and EPZS. It is also noted that for frames 243–267, PARPLMS performs up to 2 dB better than FS. Our algorithm is more effective when estimating large and complex motion due to its use of the extended rood pattern search and a moving search window. For the high-motion blocks, a large number of blocks (7–45%) are encoded using the intramode, therefore causing the predicted MV to be 0. However, in frames with high and complex motion, the optimal MV is most likely to be located outside the search window, centered at the predicted MV. In such cases, PARPLMS is able to overcome this problem by adaptively shifting the search window to a better search center, which is closer to the optimal MV while the extended rood search is used to locate the large MV.

More comprehensive test results are summarized in Tables 3 and 4. The results for the low-motion sequences in
Table 3 show that the performance for all FME algorithms is similar. In comparison to other FME algorithms, ARPS-2 has the lowest encoding and ME computations, but at the price of lower picture quality. Although UMHexagonS, EPZS, and PARPLMS results in high video quality, PARPLMS is much faster than UMHexagonS and EPZS, with a corresponding savings of 10–15% for motion estimation–related computations (or ~6% savings for the total encoding time).

For medium- and high-motion sequences, the performance of PARPLMS is comparable to FS, as shown in Tables 3 and 4. The quality of the sequences encoded using PARPLMS is about ±0.1 dB, while the variation in encoding bit rate is also within ±3%, compared to FS. PARPLMS is able to achieve a reduction of >90% in computational time over FS for high-motion sequences. Compared to UMHexagonS and EPZS, our PARPLMS algorithm carries out far fewer motion estimation–related computations (about 15–40% less), which translates to a savings of about 5–25% in the total encoding time (see Fig. 17).

6 Conclusion

In this paper, we introduce PARPLMS, which aims to improve the performance of the H.264 video coding. PAR-
PLMS enables complex and large motion to be searched effectively, despite having a low computational complexity. Our algorithm performs comparably to the FS (on average to within 0.1 dB) while having better or similar quality performance as UMHexagonS and EPZS, which are recommended for the JVT/H.264 standard. PARPLMS’s extended rood search and adaptive moving search window method enables it to produce better visual quality when estimating large and complex motion frames. Our algorithm is faster than FS by about 76–96% and up to 40% faster than UM-HexagonS and EPZS, in terms of the motion estimation computations. The use of the MV predictors, shrinking ARPS-2 method enables only the relevant points to be searched. The proposed scheme is particularly useful for low bit-rate coding because of the bigger interframe motion due to the resultant low FR.

Acknowledgments

The authors thank Dr. Lin Weisi of Institute of Infocomm Research Singapore for his valuable suggestions.

References


Hong Yin Lim studied electrical engineering at the National University of Singapore from 1999 to 2003. He is currently pursuing his PhD under the supervision of Prof. Ashraf A. Kassim at the same university. His research interests include video coding, signal processing, and multimedia.

Ashraf A. Kassim received his BEng (First Class Honors) in electrical engineering from the National University of Singapore (NUS) in 1985. From 1986 to 1988, he worked on machine vision systems at Texas Instruments. He went on to obtain his PhD in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, in 1993. Since 1993, he has been with the Electrical & Computer Engineering Department at NUS, where he is currently the vice-dean. Kassim’s research interests include image analysis, machine vision, video/image processing, and compression.
Peter H. N. de With graduated in electrical engineering from the University of Technology in Eindhoven and received his PhD from the University of Technology Delft, The Netherlands in 1992. He joined Philips Research Labs Eindhoven in 1984, where he became a member of the Magnetic Recording Systems Department. In 1996, he became senior TV systems architect and, in 1997, he was appointed as full professor at the University of Mannheim, Germany, with the faculty of computer engineering. In Mannheim, he was heading the chair on Digital Circuitry and Simulation with the emphasis on video systems. Between 2000 and 2007, he was with LogicaCMG (now Logica) in Eindhoven as a principal consultant. Early 2008, he joined CycloMedia Technology, The Netherlands, as vice-president for video technology. Since 2000, he is a professor at the University of Technology Eindhoven, with the faculty of Electrical Engineering, and he leads a group on video coding and architectures. de With is a Fellow of the IEEE, program committee member of the IEEE CES, ICIP, and VCIP, board member of the IEEE Benelux Chapters for Information Theory and Consumer Electronics, co-editor of the historical book of this community, former scientific board member of LogicaCMG, scientific advisor to Philips Research, and of the Dutch Imaging School ASCII, IEEE ISCE, and board member of various working groups.