Signal Processing: *Image Communication* 23 (2008) 725–738

Contents lists available at ScienceDirect

Signal Processing: *Image Communication*

journal homepage: www.elsevier.com/locate/image

New pixel-decimation patterns for block matching in motion estimation

Avishek Saha [a], Jayanta Mukherjee [a], Shamik Sural [b,*]

[a] *Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, WB 721302, India*
[b] *School of Information Technology, Indian Institute of Technology, Kharagpur, WB 721302, India*

ARTICLE INFO

Article history:
Received 14 July 2007
Received in revised form
31 March 2008
Accepted 25 August 2008

Keywords:
Fast motion estimation
Pixel decimation
Boundary-based decimation
N-queen
Genetic algorithm

ABSTRACT

This paper presents a boundary-based approach towards pixel decimation with applications in block-matching algorithms (BMAs). The proposed approach is based on the observation that new objects usually enter macroblocks (MBs) through their boundaries. The MBs are selected based on boundary region matching only. The boundary-based patterns can be used to speed up motion estimation with marginal loss in image quality. Different decimation levels for image quality trade-off with computational power have been presented. The mathematical intuition in support of the proposed patterns has been discussed. Apart from the boundary-based approach, the novelty in our contribution also lies in performing a genetic algorithm (GA)-based search to find optimal *M*-length patterns in an $N \times N$ block. The resultant patterns are found to have better values of spatial homogeneity and directional coverage metrics, as compared to the recently proposed *N*-queen decimation lattices. Subsequently, we obtain new pixel-decimation patterns by combining the proposed boundary-based patterns with *N*-queen patterns and the GA-based patterns. Experimental results demonstrate considerably improved coding efficiency and comparable prediction quality of these new patterns as compared to existing decimation lattices.

1. Introduction

Motion-compensated transform coding is an integral part of almost all well-known video compression standards, such as ISO MPEG1-4 [22–24], ITU-T H.261/263 [10,11] and H.264 [12]. Motion estimation (ME) achieves compression by exploiting the high temporal redundancy in successive frames of a video sequence. However, in the past few years, multimedia mobile services have witnessed a manifold growth. This has resulted in a strong demand for efficient implementation of image and video applications in next generation wireless multimedia systems. Limited processing power, battery life and memory capacity require low complexity video encoders. ME, being the most computationally intensive module, is an ideal candidate for optimization. Moreover, in portable mobile multimedia applications, the best image quality may not always be required. Therefore, algorithmic/architectural approaches may be designed to trade off image quality with encoder complexity.

Many fast algorithms have been proposed to reduce the computational complexity of ME and can be divided into six categories [14], namely, (a) reduction in search position [32,27,13], (b) simplification of matching criterion [1,20,4,30,31], (c) bitwidth reduction [21], (d) predictive search [3], (e) hierarchical search [18] and (f) fast full search [2]. The first category tries to improve ME by reducing the number of search points. The second

* Corresponding author. Tel.: +91 3222 282330; fax: +91 3222 282206.
E-mail addresses: avishek.saha@gmail.com (A. Saha), jay@cse.iitkgp.ernet.in (J. Mukherjee), shamik@sit.iitkgp.ernet.in, shamik@iitkgp.ac.in (S. Sural).

0923-5965/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.image.2008.08.004

category, also known as pixel decimation, tries to reduce the number of representative pixels selected from a block to evaluate some matching criteria. In bitwidth reduction technique, pixels are transformed into a lower resolution representation before the conventional ME search strategies are applied. Predictive search, hierarchical search and fast full search (FS) comprise the remaining three fast motion estimation techniques.

The pixel-decimation technique can be easily combined with any of the aforementioned approaches and hence forms the focus of this discussion. Initially, Bierling [1] used an orthogonal sampling lattice with a uniform 4:1 subsampling pattern to select pixels for the search of motion vectors. This scheme resulted in reduced accuracy motion vectors since the fixed pixel pattern always left out some pixels. The aliasing effects were overcome by low-pass filtering. Another scheme to remove these aliasing effects was proposed [20] which ensured that all pixels in the current block are used in the evaluation of block match by selecting one out of four different alternating 4:1 subsampling patterns in each step. However, better coding efficiency can be achieved by using adaptive patterns as compared to fixed patterns [1,20] but with an additional overhead of selecting the most representative pattern. Compared to random 4:1 subsampling, Chan and Siu [4] presented a more efficient local pixel-decimation scheme which divides a block into several regions and selects more number of pixels from regions containing higher details or edges. The concept of adaptivity was extended from local to global by suggesting [30] an algorithm that directly looks for edge pixels in 1-D space with the help of Hilbert scan. This algorithm does not require an initial division of a block and selects pixels only when they have the features important in determining a match. Experimental results show that the proposed global adaptivity is more efficient than the existing locally adaptive techniques. Pixel difference classification (PDC) [8], Minimax criterion [6], boundary match [5] and integral projection [16,26] are other pixel-decimation techniques proposed in the literature. Interestingly, it was observed that the N-queen patterns [28] outperform all other existing lattices in terms of speed and reconstructed video quality.

This paper presents new pixel-decimation patterns for block matching applications in ME. Initially, a boundary-based approach has been proposed based on the observation that, new objects enter a macroblock (MB) through its boundary regions. The aim is to capture the entry of new objects at the boundary region itself. In the proposed approach, the best matched block is selected based on partially computed boundary-region block-matching sum. A mathematical model to compute these partial block-matching sums has been presented to reasonably justify the proposed hypothesis. Apart from the boundary-based approach, the novelty of the presented work also lies in searching for optimal M-length decimation patterns in an $N \times N$ block. Subsequently, the optimal M-length patterns have been combined with the proposed boundary-based decimation patterns and the N-queen patterns. Experimental results show an improvement of 8–10 times in speed when compared to FS using the full pattern and about 1.25–2.25 times when compared with the state-of-the-art N-queen patterns. When combined with the kite cross diamond search (KCDS) [17] fast search strategy, the obtained speed improvements is in the range of 900–1900 times over the FS using the full pattern.

The rest of this paper is organized as follows. Section 2 proposes the boundary-based approach towards pixel decimation. The N-queen patterns and the genetic algorithm (GA)-based decimation lattices have been presented in Sections 3 and 4, respectively. In Section 5, we explore the effect of combining the boundary-based patterns with N-queen and GA-based lattices. Finally, in Section 6 we conclude this paper and provide future directions.

2. Boundary-based pixel decimation

This section presents a boundary-based pixel-decimation approach for reducing the computational complexity of the ME module. Among the six aforementioned categories, the pixel-decimation approach can be combined with any other category to yield further reduction in computation. Hence, this work adopts the pixel-decimation technique for complexity reduction purposes.

2.1. Proposed approach

Camera motion of a video being encoded is usually translational in nature. For a particular MB in the ith frame, any new object that is visible in the same MB in the $(i+1)$th frame, enters the MB through one of its four boundaries. The aim is to catch the motion of the new object entering the MB in the boundary of the MB itself. It is to be noted that this proposition is particularly true for cameras in mobile phones and handheld devices. Motion sequences recorded by these cameras are usually smooth. Sudden appearance/disappearance of objects at the center of the MB (without changing the boundary layers) is rather unlikely.

To detect any new object entering the concerned MB, we inspect the values of the matching criteria for boundary rows and columns only. The conjecture is that, the complete matching criteria need not be calculated and a decision can be taken based on the first few boundary-based partial computations only. The aim is to maintain the quality of partial matching based encoded sequence as close as possible to the sequence encoded with the full matching criteria evaluated over all the pixels. This provides the necessary motivation to study the effect of video reconstruction from a compressed sequence encoded with boundary-based partial matching criteria. A number of matching criteria are available in literature, namely, sum-of-absolute-differences (SAD), MAD, MAE, MSE, etc. Among these, SAD and MAE are the most popular ones. Due to the inherent annular-strip like structure of the proposed boundary-based decimation patterns, the outer boundary layers have to contain more number of pixels and we want to take into account the number of pixels in each layer. For this particular situation, MAE might not be the correct metric as compared to SAD. Using MAE would lead to loss of

information regarding the number of pixels in each layer. Hence, SAD is considered to be a more suitable matching criterion. The fact that the outer layer contains more pixels increases the probability of the max partial SAD being present in one of the boundary layers and thus further strengthens our boundary-based proposition.

2.2. Partial SAD sum features

Let the pixels of an MB be visited following a spiral scan order from the boundary to the interior as shown in Fig. 1(a)–(e). The sequence of visited pixels can be positioned by different layers, where pixels of each layer lie at the same distance from the boundary (considering the chess board distance as a metric). For example, at the ith pixel layer, the pixels should have a chess board distance from the boundary as $(i+1)$, where $0 \leqslant i < \lfloor N/2 \rfloor$. The ith layer is visited starting from the pixel at (i, i) position, situated on the diagonal of the MB. Let j denote the position of the pixel in the sequence of ith layer, where $0 < j \leqslant 4(N-i-1)$. Let the pixel at the ith layer and the jth sequential position of that layer for the $N \times N$ MB X and the $N \times N$ MB Y be denoted by x_{ij} and y_{ij}, respectively.

In the proposed computational model, the spiral scan-based partial computation of SAD values have been considered. Let a partial SAD sum at the ith layer be denoted as

$$S_i = \sum_{j=0}^{n_i} |x_{ij} - y_{ij}| \tag{1}$$

where $n_i = 4(N-i-1)$ and $0 \leqslant i < \lfloor N/2 \rfloor$. Hence, the full SAD sum in terms of S_i can be expressed as shown below:

$$\mathrm{SAD}(X, Y) = \sum_m \sum_n |x_{mn} - y_{mn}| = \sum_{i=0}^{\lfloor N/2 \rfloor - 1} S_i \tag{2}$$

where (m, n) are usual array indices.

The intention is to analyze the properties of the sequence of partial SAD sums, $\{S_0, S_1, S_2, \ldots, S_M\}$, relevant to the proposed computational approach. First, some terminologies and notations used in the subsequent discussion have been defined.

Definition 1. A sequence $S_0, S_1, S_2, \ldots, S_M$ is a monotonically decreasing sequence (MDS) if

$$S_0 > S_1 > S_2 > \cdots > S_M \tag{3}$$

Definition 2. A sequence $S_0, S_1, S_2, \ldots, S_{M-1}$ is p-MDS if

$$S_0 > S_1 > S_2 > \cdots > S_{p-1} \tag{4}$$

and $\forall j \in [p, M-1]$, $S_{p-1} > S_j$ and it is not $(p+1)$ MDS.

It may be noted that

(1) the above definitions also include 0-MDS, which implies that any finite sequence of real values of length M could be transformed into one of these $(M+1)$ sets of p-MDS, $p \in [0, M]$. On the other hand, an M-MDS (for a sequence of length M), is an usual MDS.
(2) ith element of a p-MDS $(0 \leqslant i < p)$ is the $(i+1)$th maximum. Naturally, for 0-MDS, the maximum element occurs at $i \neq 0$.

Experiments have been performed on a number of sequences, namely—*Carphone*, *Container*, *Foreman*, *Stefan*, *Tennis*, *Garden*, to observe the monotonically decreasing (MD) properties of partial SAD sequences obtained from the MBs. For each sequence, the SAD computation has been performed on blocks of previous frame only. Since the aim is to find the best possible match for the candidate block (current block in current frame), the exhaustive FS strategy has been used. Once the best match has been found using FS, the individual boundary-wise partial SADs are computed to construct a p-MDS. Thereafter, the newly

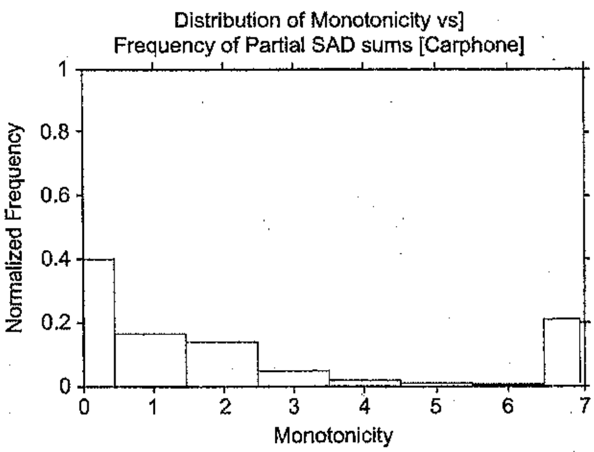

Fig. 2. Position of maximum partial SAD sums.

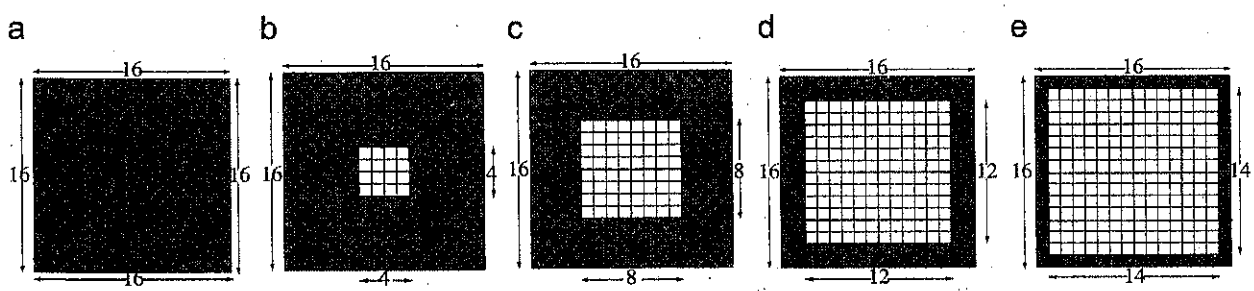

Fig. 1. Spiral scan based pixel-decimation calculation for (a) $k=8$ or no pixel decimation, (b) $k=6$, (c) $k=4$, (d) $k=2$, and (e) $k=1$.

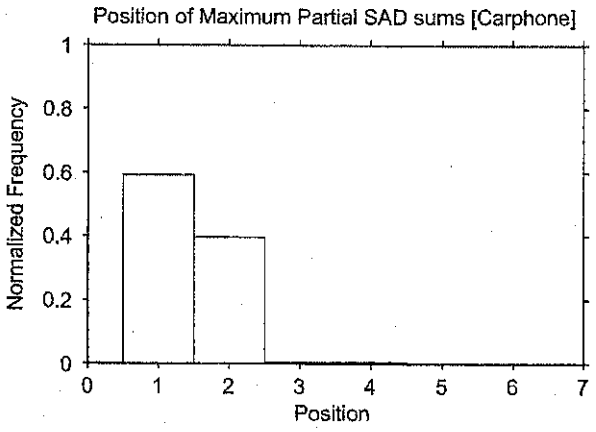

Fig. 3. Position of the maximum values in the 0th monotonic sequence.

obtained p-MDS is inspected to locate the position of the maximum partial SAD in the p-MDS. Fig. 2 shows the normalized frequency distribution of the occurrence of p-MDSs for the sequence *Carphone*, where, $p = 0, 1, 2, \ldots, \lfloor N/2 \rfloor - 1$.

It can be seen in Fig. 2 that, about 40% of the sequences are 0th-monotonic and the monotonicity of the sequences gradually decreases except in the last case. Since 40% of the sequences are 0th-monotonic, in the remaining 60% cases, the maximum partial SAD sum lies in the first column, i.e., in the outermost layer. Fig. 3 shows the distribution of the partial SAD values for 0th-monotonic sequences only.

It is to be noted that for 0-MDS sequences, the maximum partial SAD value lies in the first 2 or 3 columns. As is evident from Fig. 3, the chances of the subsequent columns (from 3 to 7) containing the maximum partial SAD value are extremely low. The possibility of the max partial SAD sum to be contained in the first 3 columns (0, 1 and 2) is

$$0.6 + (0.58 * 0.4) + (0.39 * 0.4) = 0.988 \quad (5)$$

or 98.8%. The results shown here were obtained from the test sequence *Carphone*. Similar results were obtained from tests carried out on other standard test sequences, namely *Container, Foreman, Stefan, Tennis, Garden*.

Let X and Y be any two partial SAD sum sequences of length P, i.e., $X = \{x_0, x_1, x_2, \ldots, x_{P-2}, x_{P-1}\}$ and $Y = \{y_0, y_1, y_2, \ldots, y_{P-2}, y_{P-1}\}$.

Then $|X|_k$ and $|Y|_k$ can defined as

$$|X|_k = \sum_{i=0}^{k} x_i, \quad |Y|_k = \sum_{i=0}^{k} y_i \quad (6)$$

Again, $X > Y$, iff $|X|_p > |Y|_p$. Let p_k be the probability of $X > Y$ given $|X|_k > |Y|_k$. Then p_k can expressed as follows:

$$p_k = \text{Prob}\left(|X| > |Y| \middle| \sum_{i=0}^{k} x_i > \sum_{i=0}^{k} y_i \right) \quad (7)$$

for $k = 0, 1, 2, 3, \ldots, P$, where $P = \lfloor N/2 \rfloor$.

Experiments have been carried out to estimate the probability p_k for different values of k. The data for the test have been generated in a manner such that all the partial SAD sums in the set X belong to the current MB and all the partial SAD sums in the set Y belong to the reference MB. Every partial SAD sum in the set X is compared with only those partial SAD sums in Y, which belong to the search window where the block from X would have been searched for the best match. These comparisons have been made for all possible values of k. Probability estimation was carried out on standard sequences, namely *Container, Carphone, Foreman, Stefan, Tennis, Garden*. The sequences have been selected so as to represent varying degrees and complexities of motion content. The discrete probability distribution (p_k) of all these sequences have been presented in Table 1.

Table 1
p_k for different k-values

Input sequence	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$
Container	0.76	0.78	0.84	0.96	0.92	0.96	0.98	1.00
Carphone	0.70	0.75	0.78	0.82	0.87	0.92	0.98	1.00
Foreman	0.68	0.73	0.77	0.80	0.85	0.91	0.97	1.00
Stefan	0.65	0.71	0.76	0.80	0.83	0.92	0.96	1.00
Tennis	0.63	0.72	0.74	0.78	0.81	0.89	0.95	1.00
Garden	0.67	0.75	0.78	0.81	0.84	0.92	0.95	1.00
Average p_k	0.682	0.74	0.778	0.828	0.853	0.957	0.965	1.0

The results in Table 1 show that the probability of making a correct decision based on partial SAD sum is quite high. Higher the value of k, higher is the chance of selecting the optimal block. It is to be observed in Table 1 that, at low k values p_k is very high, particularly for slow-moving sequences. In almost all the sequences, after $k = 6$, there is only marginal improvement in the probability. So, a k-value of 6 or 7 can be used for most practical purposes. $k = 7$ examines $(256 - 8 - 8 - 6 - 6) = 228$ and $k = 6$ examines $(228 - 7 - 7 - 5 - 5) = 204$ pixels, which is much less compared to the full 256 pixels. Lower values of k can also be selected depending on the amount of image quality loss or the increase in bitrate that can be afforded. Thus, this technique gives an added flexibility in selecting the k-level based on a trade-off between quality and time. Also, it is to be noted that, the focus of this paper is not the boundary-based pattern alone, but a successful merger of the boundary-based pattern with the existing N-queen patterns leading to further improvement.

2.3. Results

The proposed boundary-based pixel-decimation patterns were implemented on a typical motion estimation software (MEPackage) without any encoding and on the H.264[1] JM 10.2 reference software [15]. Table 2a shows the

[1] Encoder parameter configuration: high profile level 3.3, period of I-frames = 10, quantization parameter for I and P slices (0–51) = 28, no frames skipped, subpixel ME disabled, number of previous references frame = 2, only InterSearch 16 × 16 enabled, no B-frame used, SP-picture periodicity disabled, entropycodingmethod = CABAC, RD-optimized mode decision = 1, initial QP for rate control = 24.

Table 2
Comparative performance of sampling lattices [P:PSNRY (in dB), ΔP : ΔPSNRY (in dB), SUF: SpeedUp Factor]:

(a) *Results on MEPackage*

Pattern	Full search			KCD search		
	PSNR	ΔPSNR	SUF	PSNR	ΔPSNR	SUF
(a) *QCIF Container*						
Bbk8	43.092155	–	1.00	42.959949	–	189.65
Bbk6	43.041370	0.050785	1.07	42.904133	0.055816	203.79
Bbk4	43.020287	0.071868	1.33	42.893906	0.066043	254.60
Bbk2	42.995621	0.096534	2.29	42.891651	0.068298	433.55
Bbk1	42.979317	0.112838	4.28	42.873730	0.086219	805.79
(b) *QCIF Foreman*						
Bbk8	31.842520	–	1.00	29.741817	–	91.29
Bbk6	31.511059	0.331461	1.07	29.563042	0.178775	102.20
Bbk4	31.153343	0.689177	1.33	29.401083	0.340734	129.44
Bbk2	30.826757	1.015763	2.29	29.286697	0.455120	221.04
Bbk1	30.616280	1.226240	4.28	29.227472	0.514345	411.98

(b) *Results on H.264*

Input	Parameters	Method	P	ΔP	SUF
Foreman QCIF	±16 10 Hz 112 kbps	Bbk8	36.07	–	1.00
		Bbk6	36.06	0.01	1.07
		Bbk4	36.01	0.06	1.33
		Bbk2	35.86	0.21	2.29
		Bbk1	35.67	0.4	4.28
Foreman QCIF	±16 30 Hz 1 Mbps	Bbk8	42.64	–	1
		Bbk6	42.63	0.01	1.07
		Bbk4	42.59	0.05	1.33
		Bbk2	42.51	0.13	2.29
		Bbk1	42.42	0.22	4.28

results of various sampling lattices for two QCIF standard test sequences, namely *Container*—a slow moving sequence and *Foreman*—a moderately fast moving sequence. The results of Table 2a have been shown for both the FS strategy and a fast search strategy, namely, KCDS [17]. Table 2b compares the performance of different sampling lattices on the H.264 encoder using the FS strategy. Comparisons are made for the proposed boundary-based patterns with $k = 8$ (Bbk8), $k = 6$ (Bbk6), $k = 4$ (Bbk4), $k = 2$ (Bbk2) and $k = 1$ (Bbk1), at a low bitrate of 112 kbps and at a high bitrate of 1 Mbps. In Table 2a, the column *method* denotes the combination of search strategy and pixel decimation used. In Table 2b, all the pixel-decimation patterns have been used with the FS strategy. In both Tables 2a and b, the column P represents the PSNR value in dB, the column ΔP denotes the fall in PSNR value for a particular pixel decimation with respect to the Full sampling (Full) lattice, and the column SUF denotes the SpeedUp Factor obtained by using the proposed pixel-decimation patterns over the Full sampling (Full) lattice.

The MPEG committee uses an informal threshold of 0.5 dB PSNR to decide whether to incorporate a coding optimization [25]. It is evident from Table 2a that, the proposed boundary-based patterns perform well for both slow and fast moving sequences. The results are particularly encouraging for KCDS, where the quality loss is small but the speedup is substantial with a maximum SUF of 805.79 obtained using the Bbk1 patterns on the *Container* sequence. The only noticeable degradation in quality is for the fast moving sequence with the FS strategy. Table 2b shows the results of the proposed patterns on H.264 at low bitrates of 112 kbps and at high bitrates of 1 Mbps. As can be seen, for low bitrates, the PSNR drop of the worst-case pixel-decimation pattern, i.e., Bbk1 is 0.4 dB which is well within the informal threshold prescribed by the MPEG committee. All other patterns have a lower PSNR drop and results in substantial amount of speedup when compared to the no pixel decimation (Bbk8) case.

As is evident from the tables, at low bit-rates the PSNR loss between $k = 8$ and 6 is less than 0.5 dB. So $k = 6$ or 7 is the preferred choice of partial SAD sum computation for most practical purposes. For applications aimed at mobile devices where the best quality may not always be required, $k = 5$ and 4 may be considered. Typical mobile applications operate in the range of 10–15 fps. The proposed decimation patterns have acceptable prediction quality for low and high bitrates and at both low and high frame rates. Thus, this approach is well suited for low frame rate mobile applications. It may be noted that, the plots for $k = 1$ and 2 are very close to each other. So given a choice, it is always better to select a k-value of 1 over a k-value of 2, as it incurs almost the same amount of loss in image quality but consumes much less power. The fall in power consumption is due to the reduced number of addition operations. Let $C(k)$ denote the number of operations (additions/subtractions) required for different

Table 3
Comparative study of the proposed SAD schemes

k-Value	Subtractions	Additions	Total
8	256	255	511
6	240	239	479
4	192	191	383
2	112	111	223
1	60	59	119

values of k. $C(k)$ can be expressed as a function of k as shown below:

$$C(k) = [16 * k + (16 - k) * k] + \{[16 * k + (16 - k) * k] - 1\} \quad (8)$$

From Eq. (8), the total number of additions and subtractions required for any value of k can be computed. Table 3 makes a comparison of the number of additions and subtractions required for the proposed approaches.

Fig. 4(a) and (b) shows the reduction in operations and the reduction in image quality at different values of k. Fig. 4(c) shows the change in bitrate with changes in k, at a fixed PSNR. Software profiling of the proposed approach with $k = 1, 2, 4, 6$ and 8 was done using the *GNU gprof* profiling tool with Carphone as the test sequence. The encoding scheme was broadly divided into five functional blocks based on the amount of power consumed. The functional blocks were named as—SAD (B1), ME and compensation (B2), DCT/IDCT (B3), quantization and dequantization (B4), VLC/VLD (B5) and others (B6). All those functions which contribute insignificantly to the overall encoding time were grouped under others. Table 4 summarizes the profiling results.

The values in the table denote the percentage of execution time taken by a particular block with respect to the total execution time by the entire encoder. It can be seen that the full SAD ($k = 8$) consumes 72.28% of the total execution time. However, at lower values of k, the time taken by the partial SAD sums reduce significantly.

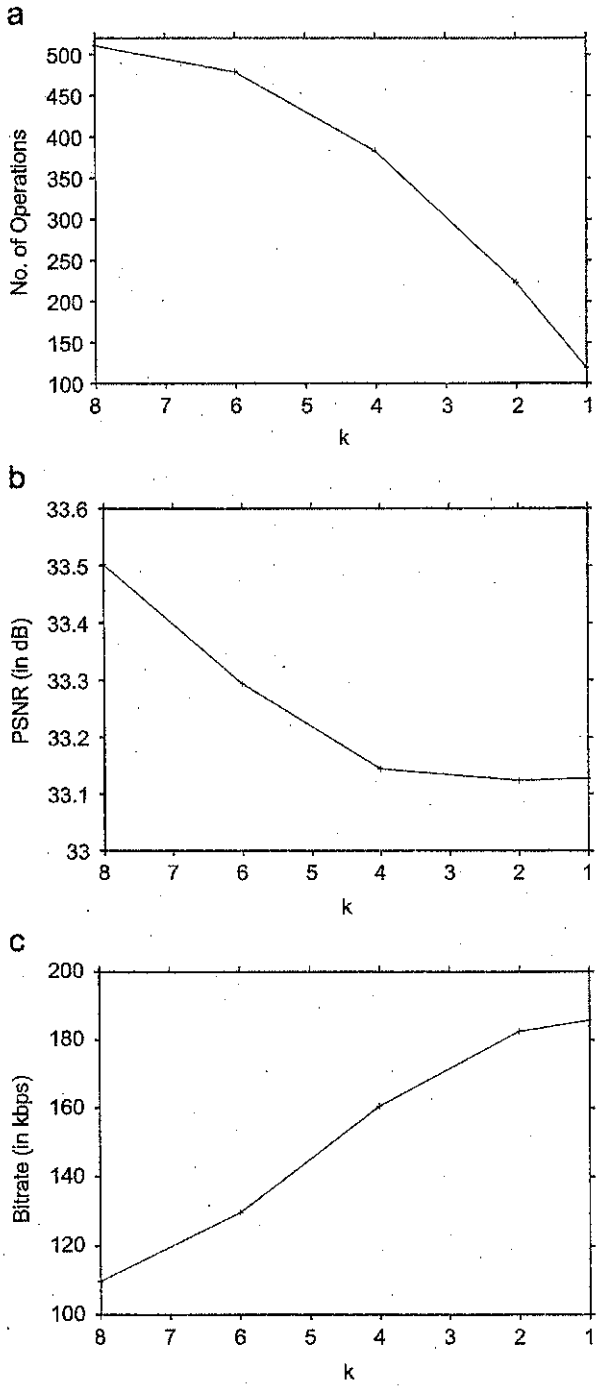

Fig. 4. For different values of k (a) reduction in number of operations at a fixed bitrate, (b) reduction in PSNR at a fixed bitrate and (c) change in bitrate at a fixed PSNR.

3. *N*-queen decimation patterns

This section gives a brief overview of the N-queen pixel-decimation patterns proposed by Wang et al. [28]. The discussion on these patterns is particularly relevant as they have been shown to out-perform other existing decimation patterns in terms of coding efficiency and picture quality.

3.1. *N*-queen approach towards pixel decimation

According to the N-queen pixel-decimation approach, the spatial information of an $N \times N$ block can be fully represented by the least number of pixels only when at least one pixel is selected from each row, column and diagonal. It has been observed that the intra-frame auto-correlation of an image is higher when the horizontal and vertical pixel spacing is close to 1. The spatial homogeneity is based on the presence of intra-frame correlations and hence is measured by the mean (μ_d) and variance (σ_d^2) of spatial distances from each skipped pixel

Table 4
Profiling results of functional blocks

K	B1	B2	B3	B4	B5	B6
8	72.28	16.85	6.77	2.93	1.45	0.32
6	66.19	21.23	7.77	4.45	0.2	0.16
4	54.9	31.36	8.83	4.31	0.21	0.39
2	40.15	41.53	11.87	5.63	0.01	0.81
1	26.85	47.48	17.9	7.75	0.01	0.01

to its nearest selected pixel:

$$\mu_d = \frac{1}{(N^2 - K)} \sum_{x=1,y=1}^{N} \|(x,y) - S(x,y)\| \quad (9)$$

$$\sigma_d^2 = \frac{1}{(N^2 - K)} \sum_{x=1,y=1}^{N} (\|(x,y) - S(x,y)\| - \mu_d)^2 \quad (10)$$

where N is the size of the block, $S(x,y)$ is the location of the selected pixel nearest to the pixel at location (x,y) and K is the number of selected pixels. Lower values of μ_d and σ_d^2 indicate higher spatial homogeneity of the sampling lattices. Directional coverage is given by the ratio of the number of edges which have at least one of the selected pixels, to the total number of edges. Edges can be lines passing through the $N \times N$ block in any of 0°, 45°, 90° and 135° directions (see Fig. 5).

3.2. N-queen results

We have implemented and tested the performance of existing pixel-decimation patterns, namely, the Quarter pattern [1], the Hexagonal pattern [7], the Quincunx pattern [19], the Yu pattern [31] and the N-queen [28]. Table 5 compares the performance of these decimation lattices. The corresponding patterns have been shown in Fig. 6. As can be seen in the table, N-queen has comparable and even better performance than the other existing decimation lattices. Detailed experimental results demonstrating better performance of the N-queen patterns can be found in Wang et al. [29].

4. GA-based decimation patterns

The N-queen approach was originally proposed for $N = 4$ and 8. However, the patterns of length other than 4 and 8 were not investigated. The question of whether there exist patterns optimal in terms of spatial homogeneity and directional coverage was also left unanswered. In addition to suggesting a boundary-based decimation approach, the novelty of our work lies in exploring new patterns of length M in an $N \times N$ block. For $N = 8, 8 \leqslant M \leqslant 16$ and $N = 16, 16 \leqslant M \leqslant 64$, the computational time required to find an optimal solution becomes prohibitively large. Hence, GA have been used to find better performing M-length patterns in an $N \times N$ block. For an 8×8 block, 8-queen uses 8 pixels and 4-queen pattern uses 16 pixels. We use GA to search for M-length patterns having better metrics of spatial homogeneity (Eq. (9) in Section 3.1) and directional coverage (Eq. (10) in Section 3.1) than N-queen lattices. Patterns of length smaller than 8 will result in poorer performance. Moreover, by definition, patterns of length more than 16 cannot improve the spatial homogeneity of an 8×8 block. The maximum value of M is taken to be 16 because it is shown that for $M = 16$, μ_d reaches its lowest value for a block of size 8×8, as discussed in the following lemma.

Lemma 4.1. *For a given block of dimension 8×8, μ_d can achieve a lowest possible value of 1, when the number of pixels selected is more than or equal to 16.*

Proof. Fig. 7 shows two decimation patterns with $M = 16$. As can be seen in Fig. 7, the lowest possible distance between every pixel and its nearest selected pixel is 1. Both the patterns in the given figure use 16 pixels and have μ_d values of 1. For sequences with $M > 16$, a new pixel position has to be considered and the new selected pixelhas to be placed in any of the unshaded location. However, it is to be noted that, this new selected pixel in no way can lower the current value of μ_d (= 1). Thus, searching for patterns with length greater than 16 is not necessary. □

Hence, GA searches for patterns of length 8–16 such that the obtained patterns have high values of spatial homogeneity and directional coverage. For similar reasons, the GA investigates patterns of length in between 32 and 64, for a 16×16 block.

4.1. GA-based pattern search

Let the number of pixels to be selected be denoted by M and the size of the block by $N \times N$. The M-length pattern selection from an $N \times N$ block can be mapped onto a simpler problem of selecting any M positions from an available set of $N \times N = N^2$ positions, such that no position in the M-length sequence occurs more than once. The aim is to select an optimal M-length pattern from the $C_M^{N^2}$ possible patterns.

A GA [9] has five components. The chromosome has been encoded as a sequence of M numbers p_i, where $0 \leqslant i \leqslant (M-1)$. No p_i is repeated in any given chromosome and all p_is have values in the range $0 \leqslant p_i \leqslant (N^2 - 1)$. The chromosome length M is an input parameter. A sample chromosome for the pattern length $M = 8$ is represented in Table 6.

This chromosome represents a decimation pattern containing the pixels in the 9th, 12th, 15th, 26th, 37th, 48th, 51st and 54th positions of an $N \times N$ block, where the numbers have been assigned to the cells of an $N \times N$ block, in a row-major fashion.

The initial set of population is an important input design parameter and in most cases is determined through experimentation. The better the input population, the faster is the convergence of the exploration. In this case, both random and improved initial input population have been used. We have tested results using population sizes of 10 000, 100 000 and 1 000 000. Initially, the chromosomes are populated in a random manner. Thereafter, successive runs of GA use improved populations from previous iterations. This work uses an elitist model [9], which preserves a few better individuals of the

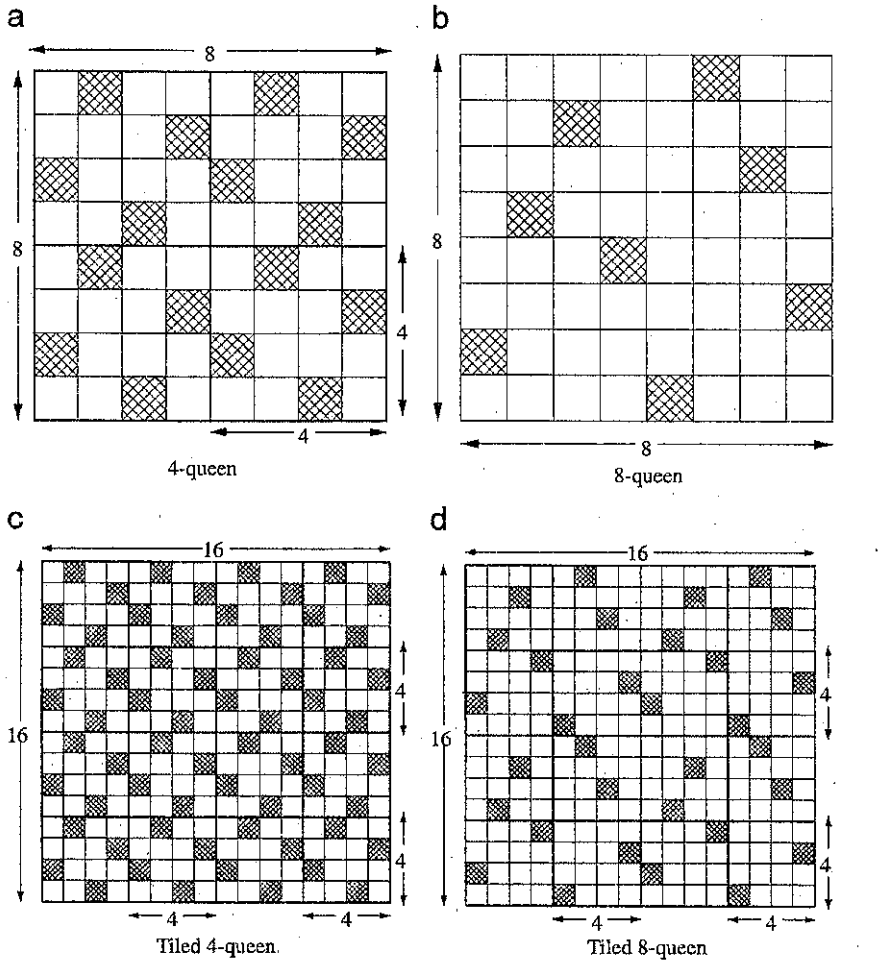

Fig. 5. *N*-queen decimation lattices.

Table 5
Comparison of the *N*-queen with other existing lattices

Input	Parameters	Method	P	ΔP	SUF
Foreman QCIF	±16 10 Hz 112 kbps	Full	36.07	–	1
		Quarter [1]	35.85	0.22	4
		Hexagonal [7]	35.87	0.20	4
		Quincunx [19]	35.99	0.08	2
		YuPattern [31]	35.84	0.23	4
		4-Queen [29]	35.85	0.22	4
		8-Queen [29]	35.6	0.47	8
Foreman QCIF	±16 30 Hz 1 Mbps	Full	42.64	–	1
		Quarter [1]	42.55	0.09	4
		Hexagonal [7]	42.58	0.06	4
		Quincunx [19]	42.62	0.02	2
		YuPattern [31]	42.57	0.07	4
		4-Queen [29]	42.58	0.06	4
		8-Queen [29]	42.47	0.17	8

previous generations in the current generation. Apart from preserving good chromosomes, we introduce few bad chromosomes from previous generation into the current generation. This helps maintain the diversity of the population and prevents the solution from getting trapped in local minima. For each iteration, 10% good chromosomes and 4% bad chromosomes from the previous generation have been preserved for the current generation. Crossover has been performed on offsprings randomly selected from the remaining 86% of the previous generation mating pool. In this application, mutation has been applied with 0.1 probability on the crossovered offsprings.

For an 8×8 block, GA tries to find out the best patterns of length M, where $8 \leqslant M \leqslant 16$. The minimum value of M is taken to be 8. Any value of M lower than 8 will result in an increase in the value of μ_d. M has a maximum value of 16 because it is shown that for $M = 16$, μ_d reaches its lowest value for a block of size 8×8, as discussed previously in Lemma 4.1.

4.2. GA-based search results

The results of the GA-based search with different pattern lengths (say M), on block sizes of 8×8 and 16×16 are shown in Table 7. As can be seen, both for $M = 8$

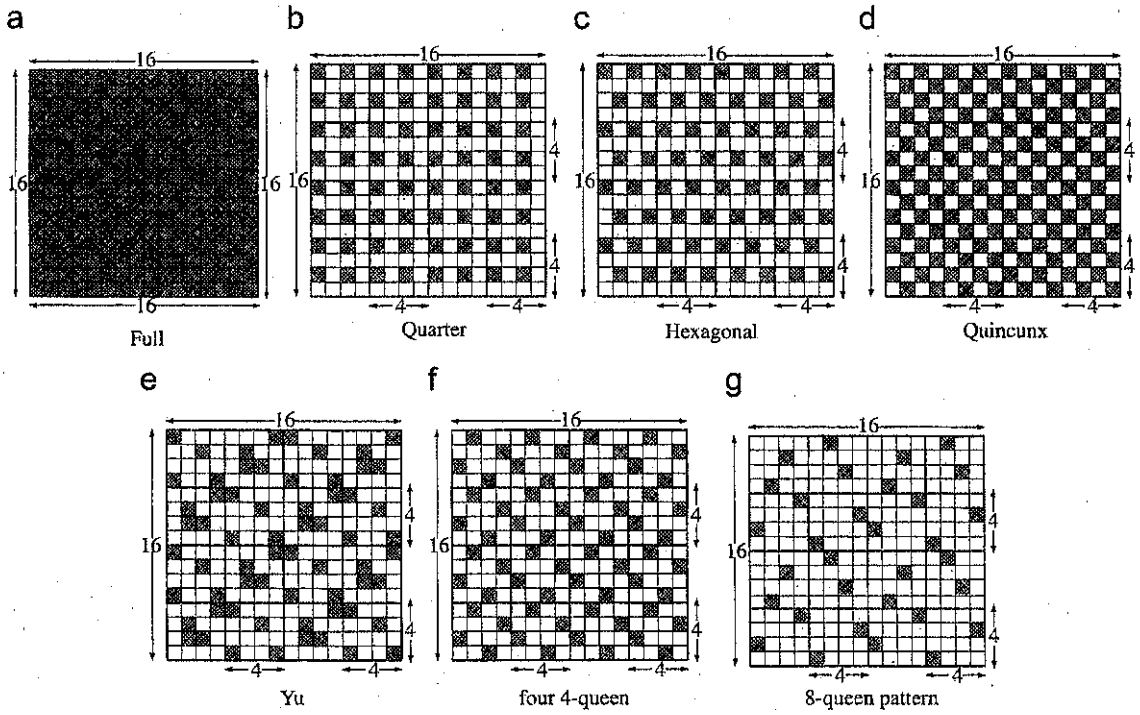

Fig. 6. Pixel-decimation patterns.

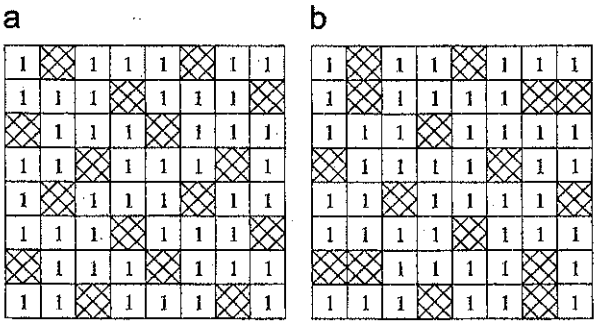

Fig. 7. 8 × 8 pixel-decimation patterns. (a) Pattern obtained by tiling four 4-queen pattern; (b) pattern obtained by GA-based search.

Table 6
Sample chromosome

9	12	15	26	37	48	51	54

and 16 for blocks of size 8 × 8 and 16 × 16, respectively, there exist patterns with μ_d much less than the 8-queen pattern. Again, GA-based $M = 16$ and GA-based $M = 64$ gives better values of directional coverage as compared to their N-queen counterparts while having similar values of μ_d. This shows that with GA-based search we can obtain pixel patterns which have better values of spatial homogeneity and directional coverage than the N-queen patterns.

The 8-queen pattern and a GA-based 8 × 8 sample pattern for $M = 8$ with $\mu_d = 1.234$ have been shown in Fig. 8a and b, respectively. By definition of μ_d, increase in the pattern length results in better values of μ_d. It can be seen that, for blocks of size 16 × 16, in most of the cases, the μ_d of a pattern is less than that obtained by tiling corresponding patterns from 8 × 8 blocks. A 16 × 16 sample pattern for $M = 32$ with $\mu_d = 1.216$ is shown in Fig. 8c. This intuitively implies that for some cases, M-length ($32 \leqslant M \leqslant 64$) patterns may result in improved performance as compared to the tiling corresponding $M/4$-length patterns obtained in the 8 × 8 case.

The experiments were performed on the H.264 JM 10.2 reference software [15]. The encoder parameters used are similar to those in Section 2.3. As discussed in Section 2, the distortion metric used was the SAD. The experiments were carried out on various M-length sampling lattices and for $N = 8$ and for 16. The sampling lattices for 16 × 16 MBs were constructed by tiling four smaller 8 × 8 sampling lattices. The experimental results on H.264 for the slow-motion sequence *Container* and the fast-motion sequence *Foreman* have been presented in Table 8. The columns P and ΔP, respectively, represent the PSNR value and the fall in PSNR value for a particular method with respect to the full sampling (FS) lattice. In the "method" column, FS_4 × 8, FS_4 × 9, etc., denote that the 16 × 16 sampling lattices were constructed by tiling four smaller 8 × 8 sampling lattices with $M = 8$, 9, and so on. For the 16 × 16 case, FS_32, FS_36, etc., denote that the 16 × 16 sampling lattices were constructed by selecting 32, 36, etc., number of pixels, respectively. For all cases, FS_16 × 4Q and FS_4 × 8Q denote that the 16 × 16 sampling lattices were constructed by tiling 16 4-queen patterns and four 8-queen patterns, respectively. SUF denotes the SpeedUp

Table 7
GA-based search results

Block size	Pattern lengths	Method	Spatial homogeneity			Directional coverage			
			μ_d	σ_d^2	σ_d/μ_d	0°	90°	45°	135°
8 × 8	8	8-Queen	1.320	0.14	28.77%	$\frac{8}{8}$	$\frac{8}{8}$	$\frac{8}{15}$	$\frac{8}{15}$
	8	GA-based $M = 8$	1.234	0.084	23.48%	$\frac{4}{8}$	$\frac{8}{8}$	$\frac{6}{13}$	$\frac{7}{15}$
	16	4-Queen tiled	1	0	0	$\frac{8}{8}$	$\frac{8}{8}$	$\frac{11}{15}$	$\frac{11}{15}$
	16	GA-based $M = 16$	1	0	0	$\frac{8}{8}$	$\frac{8}{8}$	$\frac{13}{15}$	$\frac{13}{15}$
16 × 16	32	8-Queen tiled	1.305	0.135	28.14%	$\frac{16}{16}$	$\frac{16}{16}$	$\frac{20}{31}$	$\frac{20}{31}$
	32	GA-based $M = 32$	1.216	0.079	23.11%	$\frac{12}{16}$	$\frac{14}{16}$	$\frac{19}{31}$	$\frac{23}{31}$
	64	4-Queen tiled	1	0	0	$\frac{16}{16}$	$\frac{16}{16}$	$\frac{22}{31}$	$\frac{22}{31}$
	64	GA-based $M = 64$	1	0	0	$\frac{16}{16}$	$\frac{16}{16}$	$\frac{29}{31}$	$\frac{29}{31}$

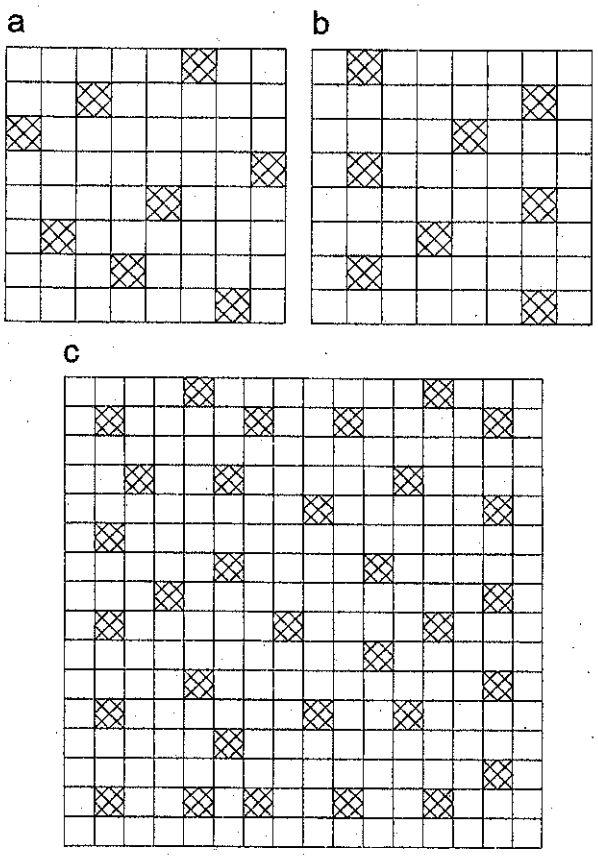

Fig. 8. Sampling lattices for (a) 8-queen based $N = 8$, $M = 8$, (b) improved GA-based for $N = 8$, $M = 8$, (c) improved GA-based for $N = 16$, $M = 32$. (a) $u_d = 1.32$; (b) $u_d = 1.234$ and (c) $u_d = 1.216$.

Factor obtained by using the proposed pixel-decimation patterns over the Full sampling (FS) lattice.

From Tables 7 and 8, the following observations can be made:

- For $M = 8$, the GA-based patterns give lower μ_d and σ_d^2 than the 8-queen pattern for both $N = 8$ and 16 ($M = 32$ pattern obtained by tiling $M = 8$ pattern). However, the 8-queen pattern gives better results in terms of PSNR when compared to the GA-based $M = 8$ pattern for $N = 16$ but not for $N = 8$. This can be seen for the *Container* sequence in Table 8. For *Foreman*, the 8-queen always gives better results as compared to the GA-based $M = 8$ pattern.
- The 4-queen tiled 16-pixel pattern and the GA-based $M = 16$ pattern yield identical μ_d and σ_d^2 values. However, it is to be noted that, the PSNR results obtained for the two cases are not identical with the 4-queen based FS_16 × 4Q pattern performing better in all the cases for both $N = 8$ and 16.
- Finally, the M-length patterns ($32 \leqslant M \leqslant 60$) for $N = 16$ have lower μ_d and σ_d^2 values than that of their corresponding $N = 8$ patterns. However, in most cases, the prediction quality of the patterns for $N = 8$ is comparable and at times even much better than that for $N = 16$.

Table 8
Results of GA-based patterns on H.264

Input file	Method	P	ΔP	Method	P	ΔP	SUF
	8 × 8 sampling lattices			16 × 16 sampling lattices			
Container	FS	30.98	–	FS	30.98	–	1
QCIF	FS_16 × 4Q	30.59	0.39	FS_16 × 4Q	30.59	0.39	4
±16	FS_4 × 16	30.55	0.43	FS_64	30.55	0.43	4
7.5 Hz	FS_4 × 8Q	30.21	0.77	FS_4 × 8Q	30.21	0.77	8
10 kbps	FS_4 × 8	30.15	0.83	FS_32	30.17	0.81	8
Foreman	FS	32.93	–	FS	32.93	–	1
CIF	FS_16 × 4Q	32.49	0.44	FS_16 × 4Q	32.49	0.44	4
±16	FS_4 × 16	32.49	0.44	FS_64	32.49	0.44	4
10 Hz	FS_4 × 8Q	31.88	1.05	FS_4 × 8Q	31.88	1.05	8
192 kbps	FS_4 × 8	31.86	1.07	FS_32	31.87	1.06	8

5. Combination of decimation patterns

The better performance of the N-queen patterns was only rationalized in terms of spatial homogeneity and directional coverage. However, the GA explorations and

analysis in Section 4 show that there exist even better patterns in terms of the prescribed criteria of spatial homogeneity and directional coverage, which do not always lead to a better performance in terms of PSNR. Thus, it can be concluded that the metrics of spatial homogeneity and directional coverage cannot give the complete information about the optimality of a sampling lattice. In addition to these criteria, there may exist some other metric which needs to be considered for a better estimate of the sampling lattice quality. This provides us the motivation to combine the GA-based patterns with N-queen and the boundary-based patterns proposed in Section 2. Fig. 9 shows some interesting results of combining the aforementioned approaches.

All the simulations have been performed on the luminance component of the popular video sequences listed in Table 9. These sequences consist of different degrees and types of motion and are in QCIF (176 × 144), CIF (352 × 288) and SIF (352 × 240) formats. The first two sequences, namely, *Container* and *Foreman*, are in QCIF

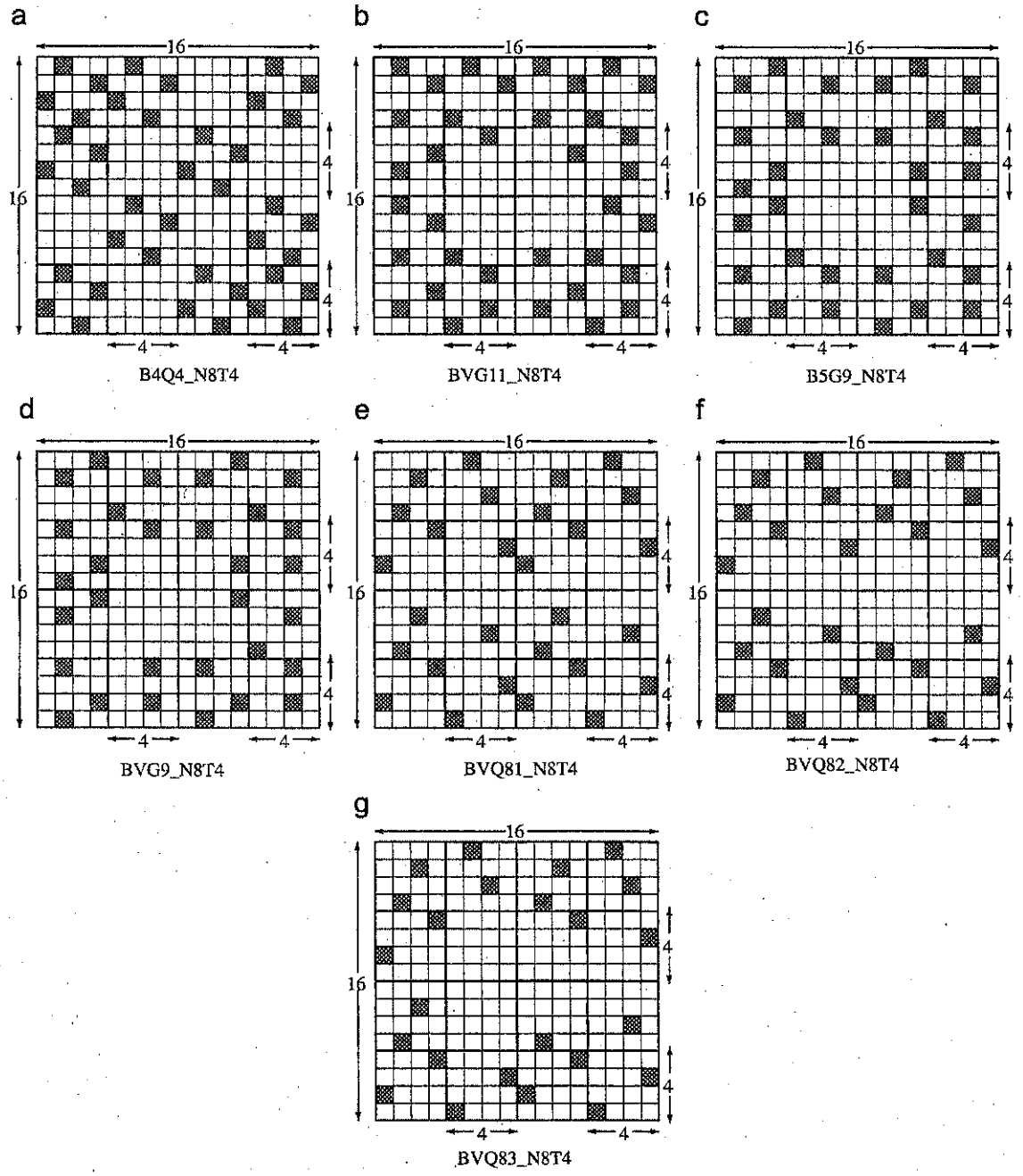

Fig. 9. Decimation patterns obtained by combining boundary-based, GA-based and N-queen-based patterns.

Table 9
Test sequences used for analysis

Sequence	Format	Total frames	Motion content
Container	QCIF (176 × 144)	299 frames	Low
Foreman	QCIF (352 × 288)	398 frames	Medium
Stefan	CIF (352 × 288)	89 frames	High
Football	SIF (352 × 240)	124 frames	High

format. The next two sequences include *Stefan* in CIF format and *Football* in SIF format. Among these sequences, *Container* has gentle, smooth and low motion change and consists mainly of stationary and quasi-stationary blocks. *Foreman* has moderately complex motion and hence is categorized as "medium" motion content sequence. Rigorous motion based on camera panning with translation and complex motion content can be found in the sequences *Stefan* and *Football*.

The search strategies used are FS for full search and KCDS for kite cross diamond search [17]. The naming notations follow the *DecimationPattern_BlockFeatures* syntax. *DecimationPattern* denotes the patterns obtained by the boundary-based (B), the *N*-queen (Q), the GA-based (G) or any combination of these three approaches. *BlockFeatures* denotes the dimensions (N) of the block for which the *DecimationPattern* has been defined and the number of tiles (T) of this block used to create the decimation pattern for a 16 × 16 block. Thus, *B6G12_N8T4* denotes that this method uses a decimation pattern obtained by combining the boundary-based $k = 6$ pattern with the GA-based decimation pattern using 12 pixels. This pattern was defined within a block of dimension 8 × 8 and four such blocks were tiled to create the decimation pattern for a 16 × 16 block. Similarly, *B4Q4G12_N8T4* denotes that the decimation pattern used is a combination of boundary-based $k = 4$ pattern, the 4-queen pattern and the GA-based $M = 12$ pattern, defined within an 8 × 8 block and tiled 4 times for a 16 × 16 block. The letters *BV* in *DecimationPattern* denote that the number of horizontal and vertical layers for the boundary-based approach are not identical.

The proposed boundary-based pixel decimation patterns were implemented on a typical motion estimation software (MEPackage) and on the H.264 JM 10.2 reference software [15]. The comparison results on MEPackage demonstrate the performance of the ME module only and do not take into account the loss incurred by the other encoding modules (like quantization, etc.). The search strategies used are FS and KCDS. The newly proposed patterns have been compared with Full (Full), Quarter (Quartr) [1], Hexagonal (Hexgnl) [7], Quincunx (Quincx) [19], Yu's (YuPatt) [31] and *N*-queen [29] patterns. For the sake of brevity, Tables 10 and 11 present results for comparison with *N*-queen patterns only.

Table 10 shows the results on MEPackage, using the slow-moving test sequence *Container*, moderately fast sequence *Foreman* and the complex motion sequences of *Stefan* and *Football*. It can be seen that, for the slow-moving sequence *Container*, the *B4Q4_N8T4* pattern performs better than most other patterns and its

Table 10
Comparative performance of the sampling lattices on MEPackage:

Pattern	Full search			KCD search		
	PSNR	ΔPSNR	SUF	PSNR	ΔPSNR	SUF
(a) *Container QCIF*						
4-Queen	43.084152	0.008003	4.02	42.927536	0.032413	751.35
B4Q4_N8T4	43.080486	0.011669	6.44	42.920422	0.039527	1197.59
BVG11_N8T4	43.042751	0.049404	7.16	42.913948	0.046001	1329.31
8-Queen	43.003960	0.088195	8.06	42.938786	0.021163	1495.78
B5G9_N8T4	43.037495	0.05466	8.06	42.937934	0.022015	1490.28
BVG9_N8T4	43.032495	0.05966	8.32	42.935885	0.024064	1537.91
BVQ81_N8T4	43.003197	0.088958	9.23	42.922260	0.037689	1706.13
BVQ82_N8T4	43.003986	0.088169	9.95	42.907482	0.052467	1838.89
BVQ83_N8T4	42.988968	0.103187	10.79	42.910309	0.04964	1991.41
(b) *Foreman QCIF*						
4-Queen	31.762175	0.080345	4.01	29.710869	0.030948	368.04
B4Q4_N8T4	31.648329	0.194191	6.44	29.626741	0.115076	591.52
BVG11_N8T4	31.538610	0.30391	7.16	29.636379	0.105438	654.06
8-Queen	31.568401	0.274119	8.06	29.615883	0.125934	740.14
B5G9_N8T4	31.511839	0.330681	8.06	29.615767	0.12605	740.14
BVG9_N8T4	31.491819	0.350701	8.32	29.602470	0.139347	764.2
BVQ81_N8T4	31.518425	0.324095	9.23	29.573584	0.168233	849.23
BVQ82_N8T4	31.468256	0.374264	9.95	29.540432	0.201385	915.07
BVQ83_N8T4	31.405787	0.436733	10.78	29.511919	0.229898	993.67
(c) *Stefan CIF*						
4-Queen	25.816923	0.081551	4.01	23.422340	0.330206	289.44
B4Q4_N8T4	25.688566	0.209908	6.43	23.150133	0.602413	464.53
BVG11_N8T4	25.650225	0.248249	7.15	23.022579	0.729967	517.11
8-Queen	25.621904	0.27657	8.05	22.916864	0.835682	586.84
B5G9_N8T4	25.611561	0.286913	8.32	22.959156	0.79339	599.77
BVG9_N8T4	25.587299	0.311175	8.32	22.915642	0.836904	600.13
BVQ81_N8T4	25.536016	0.362458	9.22	22.784330	0.968216	671.18
BVQ82_N8T4	25.483351	0.415123	9.94	22.705965	1.046581	722.13
BVQ83_N8T4	25.397606	0.500868	10.77	22.656179	1.096367	783.11
(d) *Football SIF*						
4-Queen	22.716427	0.160696	4.01	20.289766	0.120447	299.61
B4Q4_N8T4	22.470604	0.406519	6.44	20.085005	0.325208	485.81
BVG11_N8T4	22.419941	0.457182	7.16	20.031691	0.378522	540.22
8-Queen	22.407551	0.469572	8.06	20.017942	0.392271	609.36
B5G9_N8T4	22.397823	0.479300	8.32	20.027893	0.382320	625.00
BVG9_N8T4	22.346809	0.530314	8.32	19.985102	0.425111	628.92
BVQ81_N8T4	22.263395	0.613728	9.22	19.909904	0.500309	700.26
BVQ82_N8T4	22.165274	0.711849	9.94	19.845882	0.564331	756.13
BVQ83_N8T4	22.024141	0.852982	10.77	19.763897	0.646316	820.95

performance is very close to the full pattern. The speedups achieved for these patterns are about 6.44 and 1197.59, which are much more than the 4-queen patterns. It is interesting to note that the *B5G9_N8T4* and *BVG9_N8T4* patterns have lower quality loss but identical or higher coding efficiency than the 8-queen patterns. The patterns *BVQ81_N8T4*, *BVQ82_N8T4* and *BVQ83_N8T4* are the most interesting, as they provide substantial speedup in performance with very low loss in quality. Except at low bitrates where the loss due to these patterns is not sufficiently small, these patterns give satisfactory quality with substantial improvements in coding efficiency. Moreover, the worst case PSNR drop of the proposed decimation lattices for *Container* is 0.103187 dB (*BVQ83_N8T4* using FS). For these patterns, the letters *BV* denote that variable number of horizontal and vertical

Table 11
Comparative performance of the sampling lattices on H.264

Pattern	±16, 10 Hz, 112 kbps			±16, 30 Hz, 1 Mbps		
	PSNR	ΔPSNR	SUF	PSNR	ΔPSNR	SUF
Foreman QCIF						
4-Queen	35.85	0.22	4	42.58	0.06	4
B4Q4_N8T4	35.68	0.39	6.4	42.50	0.14	6.4
BVG11_N8T4	35.62	0.45	7.11	42.49	0.15	7.11
8-Queen	35.60	0.47	8	42.47	0.17	8
B5G9_N8T4	35.65	0.42	8	42.50	0.14	8
BVG9_N8T4	35.63	0.44	8.25	42.49	0.15	8.25
BVQ81_N8T4	35.49	0.58	9.14	42.43	0.21	9.14
BVQ82_N8T4	35.44	0.53	9.85	42.42	0.20	9.85
BVQ83_N8T4	35.37	0.7	10.67	42.38	0.24	10.67

layers have been used for the boundary-based pattern. This has been merged with an 8-queen pattern and the whole pattern has been defined for 8×8 blocks and tiled 4 times to construct an equivalent decimation pattern for a 16×16 block. The numbers 1, 2 and 3 denote the serial numbers of different orientations that were obtained for the aforementioned base pattern. For the KCD search, the patterns *B5G9_N8T4* and *BVG9_N8T4* have lower PSNR drop and higher coding efficiency than the 4-queen pattern. Similar observations can be made for the remaining test sequences. In all cases using the FS, the proposed patterns perform better than Quarter and Hexagonal patterns. For the fast-moving sequences *Stefan* and *Football* using the KCD search strategy, *B5G9_N8T4* gives better prediction quality and higher speedup as compared to the 8-queen pattern.

Table 11 shows the results at low and high bitrates on a typical H.264 reference encoder, for the sequence *Foreman*. At low bitrates, the *BVG9_N8T4* is the suggested pattern as it gives the highest speedup with affordable loss in quality. However, at higher bitrates, the pattern to be used is the *BVQ83_N8T4*, which gives about 10.67 times speedup with only a marginal loss of 0.24 dB. These patterns, when combined with the KCDS, result in substantial speedups of 764.2 and 963.47.

The new sampling patterns are not claimed to be the best performing ones among the exhaustive set of all possible combinations of the boundary-based, the GA-based and the *N*-queen approaches. These better performing sampling patterns are only indicative of the fact that, decimation lattices better than the *N*-queen patterns can be obtained if the existing *N*-queen patterns are combined with the proposed boundary-based approach.

6. Conclusions and future work

This paper has proposed new pixel-decimation patterns for block matching with applications in motion estimation. First, the reconfigurable boundary-based pixel-decimation patterns were introduced. These patterns were based on boundary-region partial SAD matching sums. In addition, genetic algorithm was employed to search for optimal *M*-length patterns in an $N \times N$ block. However, the highlight of this paper is the combination of the proposed boundary-based decimation patterns with *N*-queen patterns and the patterns obtained from the GA-based search. These combined patterns are interesting as they have almost identical quality loss but achieve much higher coding efficiency. At least two patterns (namely, *B5G9_N8T4* and *BVG9_N8T4*) give better PSNR results than *N*-queen while having the same or marginally better speedup. Apart from presenting better pixel-decimation patterns, the contribution of this paper also lies in the proposition that combining boundary-based approach with *N*-queen results may lead to even better pixel-decimation patterns. KCDS has been used as the underlying search mechanism for all pixel-decimation patterns to obtain the PSNR and SUF results. Our emphasis lies in the analysis of the comparative performance of various decimation patterns irrespective of the search strategy used. KCDS may be substituted by full search or any other search algorithm. While this might affect the actual execution times, the trend in speedup improvement is expected to be similar. Future work lies in proposing a more accurate optimality criteria for these sub-sampling patterns. Moreover, efficient memory addressing schemes for the proposed decimation patterns may also be looked into.

Acknowledgment

This work has been supported by a research grant from Department of Science and Technology (DST), India under Grant no: SR/S3/EECE/024/2003.

References

[1] M. Bierling, Displacement estimation by hierarchical block matching, in: Proceedings of the SPIE Conference on Visual Communications and Image Processing, vol. 1001, 1988, pp. 942–951.
[2] M. Brunig, W. Niehsen, Fast full-search block matching, IEEE Trans. CSVT 11 (2) (2001) 241–247.
[3] J. Chalidabhongse, C. Kuo, Fast motion vector estimation using multiresolution-spatio-temporal correlations, IEEE Trans. CSVT 7 (3) (1997) 477–488.
[4] Y. Chan, W. Siu, New adaptive pixel decimation for block motion vector estimation, IEEE Trans. CSVT 6 (1996) 113–118.
[5] M. Chen, Predictive motion estimation algorithms for video compression, J. St. John's St. Mary Inst. Technol. 15 (3) (1997) 197–214.
[6] M. Chen, L. Chen, T. Chiueh, Y. Lee, A new block-matching criterion for motion estimation and its implementation, IEEE Trans. CSVT 5 (3) (1995) 231–236.
[7] K. Choi, S. Chan, T. Ng, A new fast motion estimation algorithm using hexagonal subsampling pattern and multiple candidate search, in: Proceedings of the IEEE ICIP, 1996, pp. 497–500.
[8] H. Gharavi, M. Mills, Block matching motion estimation algorithms—new results, IEEE Trans. Circuits Syst. 37 (5) (1990) 649–651.
[9] D. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, Addison-Wesley, Reading, MA, 2000 (third Indian reprint ed.).
[10] H261, Video codec for audiovisual services at px64 kbit/s, ITU-T SG15, ITU-T Rec. H.261, second ed., 1993.
[11] H.263, L-T.R., Video coding for low bit rate communication, ITU-T SG16, ITU-T Rec. H.263, third ed., 2000.
[12] H.264, Joint video team (jvt) of itu-t and iso/iec jtc1, Geneva, jvt of iso/iec mpeg and itu-t vceg, jvt-g050r1, Draft ITU-T Rec. and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264-ISO/IEC 14 496-10 AVC), 2003.

[13] Y. Huang, S. Ma, C. Shen, L. Chen, Predictive line search: an efficient motion estimation algorithm for mpeg-4 encoding systems on multimedia processors, IEEE Trans. CSVT 13 (1) (2003) 111–117.
[14] Y.W. Huang, C.Y. Chen, C.H. Tsai, C.F. Shen, L.G. Chen, Survey on block matching motion estimation algorithms and architectures with new results, J. VLSI Signal Process. 42 (3) (March 2006) 297–320.
[15] JM10.2, H.J., 2006 (http://iphome.hhi.de/suehring/tml/).
[16] J. Kim, R. Park, A fast feature-based block matching algorithm using integral projections, IEEE J. Selected Areas Commun. 10 (5) (1992) 968–979.
[17] C. Lam, L. Po, C. Cheung, A novel kite-cross-diamond search algorithm for fast video coding and videoconferencing applications, in: IEEE ICASSP, 2004, pp. 365–368.
[18] J. Lee, N. Lee, Variable block size motion estimation algorithm and its hardware architecture for h.264, in: Proceedings of the IEEE International Symposium on Circuits Systems (ISCAS), 2004, pp. 740–743.
[19] K. Lengwehasatit, A. Ortega, Probabilistic partial-distance fast matching algorithms for motion estimation, IEEE Trans. CSVT 11 (February 2001) 139–152.
[20] B. Liu, A. Zaccaring, New fast algorithms for the estimation of block motion vector, IEEE Trans. CSVT 3 (1993) 148–157.
[21] J. Luo, C. Wang, T. Chiang, A novel all-binary motion estimation (abme) with optimized hardware architectures, IEEE Trans. CSVT 12 (8) (2002) 700–712.
[22] MPEG1, Information technology coding of moving pictures and associated audio for digital storage media at up to about 1.5 mbit/s part 2: video, JTC1/SC29/WG11, ISO/IEC 11 172-2 (MPEG-1 Video), 1993.
[23] MPEG2, Generic coding of moving pictures and associated audio information—part 2: video, ITU-T and ISO/IEC JTC 1, ITU Rec. H.262-ISO/IEC 13 818-2 (MPEG-2 Video), 1994.
[24] MPEG4, Information technology coding of audio visual objects part 2 visual, JTC1/SC29/WG11, ISO/IEC 14 469-2 (MPEG-4 Visual), 2000.
[25] PSNR, 2006 (http://bmrc.berkeley.edu/courseware/cs294/fall97/assignment/psnr.html/).
[26] K. Sauer, B. Schwartz, Efficient block motion estimation using integral projections, IEEE Trans. CSVT 6 (5) (1996) 513–518.
[27] A. Tourapis, O. Au, M. Liou, G. Shen, I. Ahmad, Optimizing the mpeg-4 encoder advanced diamond zonal search, in: Proceedings of the IEEE International Symposium on Circuits Systems 2000, vol. 3, 2000, pp. 674–677.
[28] C.N. Wang, S.W. Yang, C.M. Liu, T. Chiang, A hierarchical decimation lattice based on n-queen with an application for motion estimation, IEEE Signal Process. Lett. 10 (8) (August 2003) 228–231.
[29] C.N. Wang, S.W. Yang, C.M. Liu, T. Chiang, A hierarchical n-queen decimation lattice and hardware architecture for motion estimation, IEEE Trans. CSVT 14 (4) (April 2004) 429–440.
[30] Y. Wang, Y. Wang, H. Kuroda, A globally adaptive pixel decimation algorithm for block-motion estimation, IEEE Trans. CSVT 10 (6) (2000) 1006–1011.
[31] Y. Yu, J. Zhou, C.W. Chen, A novel fast block motion estimation algorithm based on combined subsamplings on pixels and search candidates, J. Visual Commun. Image Rep. 12 (2001) 96–105.
[32] S. Zhu, K. Ma, A new diamond search algorithm for fast block-matching motion estimation, IEEE Trans. Image Process. 9 (2) (2000) 287–290.