Local Binary Pattern Approach for Fast Block Based Motion Estimation

by

Rohit Verma

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Rohit Verma

Abstract

With the rapid growth of video services on smartphones such as video conferencing, video telephone and WebTV, implementation of video compression on mobile terminal becomes extremely important. However, the low computation capability of mobile devices becomes a bottleneck which calls for low complexity techniques for video coding. This work presents two set of algorithms for reducing the complexity of motion estimation. Binary motion estimation techniques using one-bit and two-bit transforms reduce the computational complexity of matching error criterion, however sometimes generate inaccurate motion vectors. The first set includes two neighborhood matching based algorithms which attempt to reduce computations to only a fraction of other methods. Simulation results demonstrate that full search local binary pattern (FS-LBP) algorithm reconstruct visually more accurate frames compared to full search algorithm (FSA). Its reduced complexity LBP (RC-LBP) version decreases computations significantly to only a fraction of the other methods while maintaining acceptable performance. The second set introduces edge detection approach for partial distortion elimination based on binary patterns. Spiral partial distortion elimination (SpiralPDE) has been proposed in literature which matches the pixel-to-pixel distortion in a predefined manner. Since, the contribution of all the pixels to the distortion function is different, therefore, it is important to analyze and extract these cardinal pixels. The proposed algorithms are called lossless fast full search partial distortion elimination ME based on local binary patterns (PLBP) and lossy edge-detection pixel decimation technique based on local binary patterns (ELBP). PLBP reduces the matching complexity by matching more contributable pixels early by identifying the most diverse pixels in a local neighborhood. ELBP captures the most representative pixels in a block in order of contribution to the distortion function by evaluating whether the individual pixels belong to the edge or background. Experimental results demonstrate substantial reduction in computational complexity of ELBP with only a marginal loss in prediction quality.

Acknowledgements

I am grateful to Dr. Mohamed-Yahia Dabbagh and Dr. Fakhreddine Karray, who have been very helpful and wonderful mentors. This work would not have been possible without their guidance and encouragement. I think the learning experience and support I received throughout my studies, kept me motivated to work and perform. I would also like to thank Dr. Zhou Wang and Dr. Hamid R. Tizhoosh for their support and valuable comments as my course instructors. They were very helpful in providing useful comments and engaged in fruitful discussions during the projects which ultimately developed into this thesis. I feel incredibly fortunate to have the privilege of interacting with such great people.

I would like to thank my friends in Waterloo for making my stay memorable, enjoyable and learning experience. My thesis would not have been possible without their support.

Lastly but not the least, I would like to thank my parents and my sister for motivating and encouraging me during the tough and distressing times during my study. I simply would not be the person that I am today without their help and unconditional love. I thank my parents for the valuable education that they have provided me and guidance through every step of my life.

Dedication

I would like to dedicate it to my parents, Rajesh & Reena, my sister Ritu and my best friend Sushweta.

Table of Contents

List of Tables

List of Figures

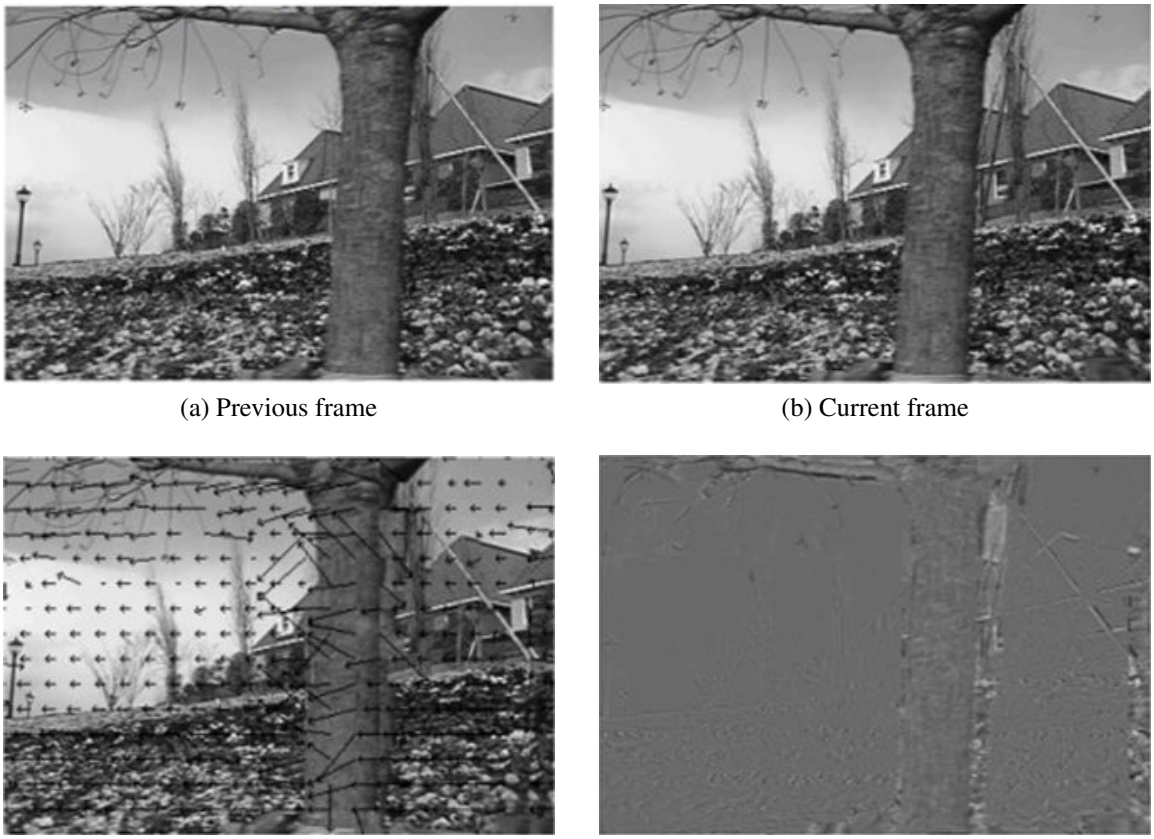
Chapter 1

Introduction

1.1 Motivation

Nowadays, the demand for video communication is increasing with the development of mobile communication systems such as videophone and digital multimedia broadcasting. 3G technologies and faster internet access enable us to use video conferencing and WebTV on mobile devices. Therefore, the limited bandwidth and computation capability demand faster video compression techniques. Motion Estimation (ME) has been widely used in many applications, but it is regarded as the most computationally intensive part [29]. Block matching algorithm (BMA) as shown in Fig. 1.1 is the simplest and most popular ME algorithm. But, its huge computational complexity makes it impractical for real-time applications. Therefore, low complexity ME algorithms with reasonable visual quality of compressed frames are required.

Several fast techniques have been proposed in literature that reduce the computational complexity of Full Search Algorithm (FSA). These include algorithms that either try to reduce the search space effectively or by reducing the number of representative pixels to be matched be-



(a) Previous frame

(b) Current frame

(c) Prediction with displacement vectors

(d) Motion-compensated prediction error

Figure 1.1: Block-based Motion Estimation procedure.

tween candidate and test blocks. There are two categories of these algorithms: lossless whose performances are the same as that of FSA and lossy which suffers from performance degradation at the cost of computational efficiency. A survey of some of these algorithms has been done in Chapter 2.

1.2 Summary of Contribution

In this work, we analyze the pixel-to-pixel distortion error between the block to be coded and the test block using local binary pattern (LBP) [44], and we propose two ME algorithms based on neighborhood matching and two ME algorithms based on edge detection.

Full search local binary pattern (FS-LBP) block motion estimation essentially captures the neighborhood information around a central pixel which is used as the matching criterion along with the pixel intensity values which has been used alone in classical sum of absolute differences (SAD) till now. It is shown that the proposed LBP based block motion estimation results in visually more accurate frames compared to Full search algorithm (FSA). It provides better performance in terms of structural similarity (SSIM) of the FS-LBP reconstructed frames. However, it is computationally intensive, therefore a reduced complexity LBP (RC-LBP) is also proposed which reduces computations significantly to only a fraction of the other methods while maintaining acceptable performance.

A lossless fast full search partial distortion elimination ME based on local binary patterns (PLBP) and a lossy edge-detection pixel decimation technique based on local binary patterns (ELBP) has also been proposed. PLBP is based on the generation of a good sequence for matching pixels using the diversity information at the local level. The matching order is derived such that the pixels which have more non-uniformity around itself contribute more to the distortion function. Experimental results show that they reduce the computations by 28% over spiral partial distortion elimination (SpiralPDE) [1]. It has also been shown that not all pixels are necessary in evaluating the SAD distortion function and that only a subset of these containing important information about the local texture of the image can be used to obtain similar results. Thus, ELBP has been proposed which extracts these cardinal pixels efficiently. It has been compared with other pixel decimation based lossy approaches, and there is a significant computational gain.

1.3 Thesis Organization

The thesis is organized as follows. Chapter 1 presents the motivation behind the work and the contributions. We discuss various algorithms categorized under three broad classes in Chapter 2. A general overview of local binary patterns is provided in Chapter 3, where we discuss the benefits of using LBP approach in motion estimation. In Chapter 4, we discuss the two neighborhood matching algorithms based on local binary patterns providing extensive analysis and simulation results. We further discuss edge detection approaches for partial distortion elimination based motion estimation in Chapter 5. We conclude the thesis in Chapter 6.

Chapter 2

General Background: Motion Estimation and Related Work

In Block matching algorithm, the current frame is divided into a matrix of macroblocks (MB) which are then compared within the MB's search range in the reference frame to come up with motion vectors (MV) according to a certain matching error criterion such as sum of absolute differences (SAD) or the sum of squared differences (SSD). The full search algorithm exhaustively searches for the minimum cost at each possible location in the search window thus giving optimal motion vectors.

The SAD of two blocks of size $N \times N$ is given in (2.1),

$$\text{SAD}(m,n) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |I^t(i,j) - I^{t-1}(i+m, j+n)| \tag{2.1}$$

where I^t and I^{t-1} are the pixel intensity values of the current and the reference image frames,

and (m, n) denotes the candidate block displacement. The motion vector is given in (2.2),

$$\text{MV} = \{(u, v) | \text{ SAD}(u, v) \leq \text{SAD}(m, n);\ -s \leq m, n \leq s\} \tag{2.2}$$

For a search range of $\pm s$, the motion vector of current block is the one with minimum SAD among the $(2s+1)^2$ search positions. As mentioned earlier, FSA has been improved by reducing the search space for candidate block (fast searching) [11, 24, 30, 32] and by reducing the number of matching pixels for SAD calculations (fast matching) [25, 53]. It is important to note that the matching process is nested into the searching process. Many new reduced computational complexity techniques have been proposed in literature. Some of them can be categorized into the reduction in search positions, the simplification of matching criterion and the bitwidth reduction techniques. The following subsection provides a brief overview of these categories. Although, many hybrid methods [22, 35] are also reported in literature which make use of a combination of these categories but basic categorization is made depending on the main characteristic of a particular approach. The following section provides a brief overview of both lossless and lossy techniques for these approaches.

2.1 Fast Searching Techniques

Fast searching approaches involve pruning the search space effectively with or without a loss in prediction performance. The key idea is to reach the optimal MV in a minimum number of searches by decimating the search space and looking into the region of low distortion.

One of the earliest algorithms that reduced the number of search positions was the Three-Step-Search (TSS) introduced by Koga [28]. It became very popular because of its simplicity, robustness and near optimal performance. It recursively searches for the best MV in a coarse

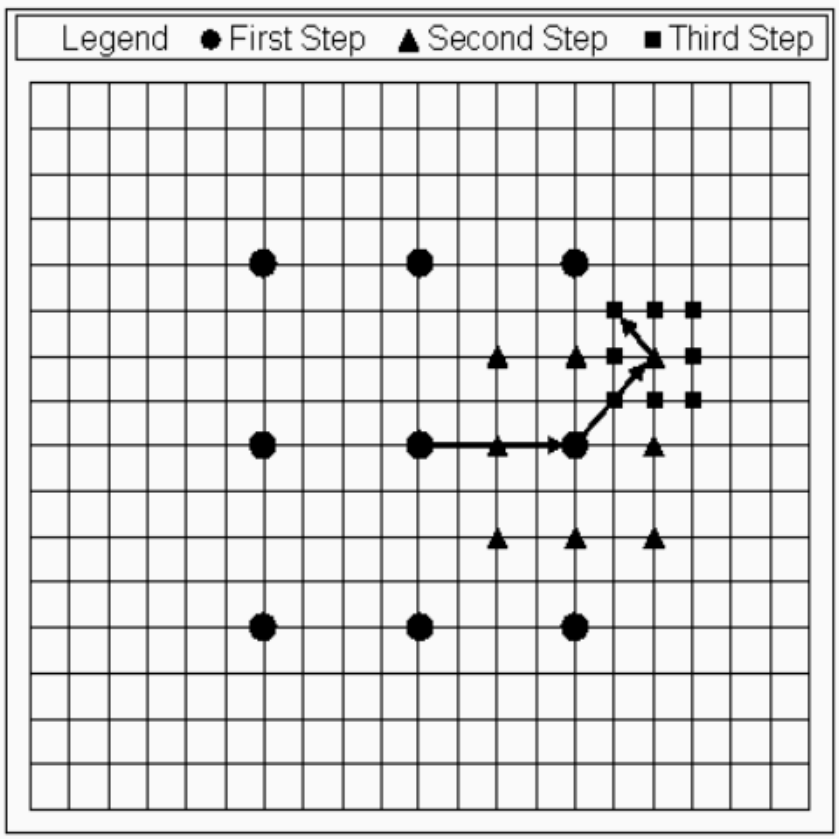


Figure 2.1: Three Step Search Procedure [3].

to fine search pattern. A sample procedure for a search range of ± 7 in a macroblock is shown in Fig. 2.1. It starts with the search location at the center and sets the step size S = 4. It then searches at the eight positions $\pm$S pixels around around the center. From these nine locations searched so far it picks the one with the least distortion cost and makes it the new search origin but now, it sets the new step size S = S/2, and then repeats in a similar fashion for two more iterations until S = 1. At that point it finds the location with the least cost and the macroblock at that location is the best match. Therefore, in this case it reduces the computational complexity by a factor of nine. The main assumption that was used in TSS is that there is a monotonic

increase in the the distortion as the search position deviates from the optimal position. Thus, it effectively computes the near-optimal MV by decimation of search positions. However, as it uses a uniformly allocated checking point pattern in the first step, it becomes inefficient for small motion estimation.

Addressing this issue, a center-biased New Three-Step-Search (NTS) [33] was proposed. The TSS uses a uniformly allocated checking pattern for motion estimation and is prone to missing small motions. In the first step sixteen points are checked in addition to the search origin for lowest cost. Of these additional search locations, eight are a distance of S = 4 away (like TSS) and the other eight are at S = 1 away from the origin. If the lowest cost is at the origin then the search is stopped and the motion vector is set to (0, 0). If the lowest cost is at any of the eight locations at S = 1, then the origin of the search point is changed to that point and checks are made adjacent to it. On the other hand, if the lowest cost after the search step was one of the eight locations at S = 4, then the normal TSS procedure is followed. Thus, it makes the search adaptive to the motion vector distribution by checking an additional eight neighbors to the center pixel.

A Four-Step-Search (4SS) [47] was also proposed which is similar to NTS. It is center-biased and has a half-way stopping technique but it uses a smaller initial step size compared to NTS. Fig. 2.2 shows a sample procedure. 4SS sets a fixed pattern size of S = 2 for the first step independent on the value of the search area. Thus it calculates the cost at the nine locations in the 5×5 window. Therefore, depending on the location of the least cost, it may either check three or five locations. Once again if the least cost location is at the center of the search window, it jumps to the fourth step or else it continues with the third step. In the final step, the search window size is set to 3×3, and search radius is dropped to 1.

Diamond Search (DS) proposed in [59] does not fix the search window size, alternatively, it employs the use of large diamond search (LDS) pattern and small diamond search (SDS) pattern

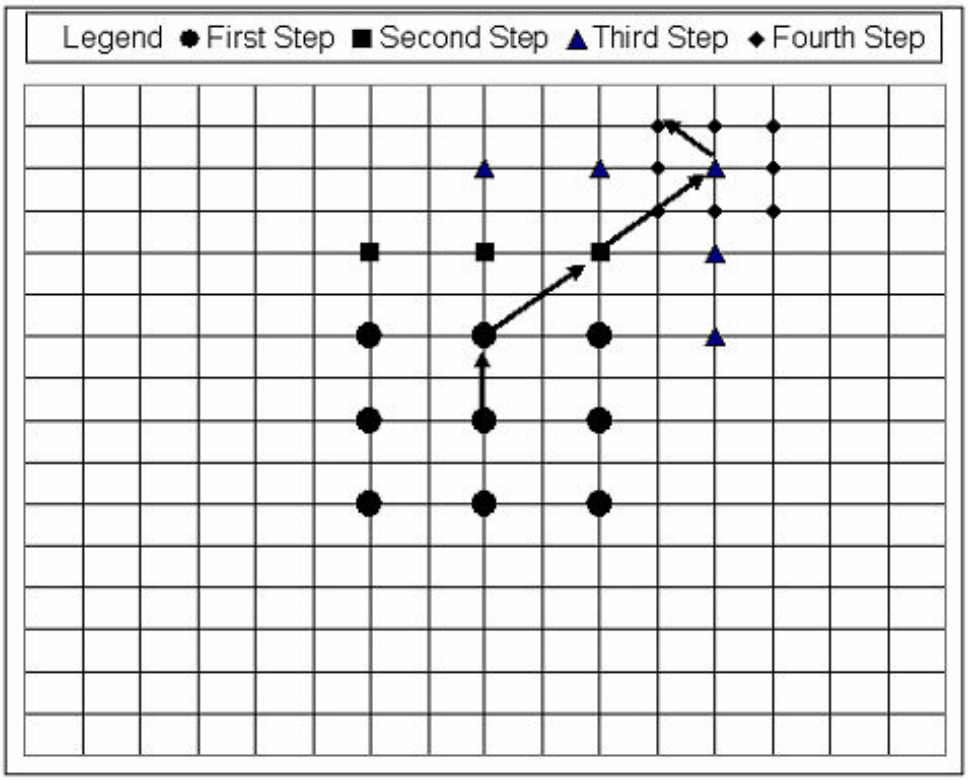


Figure 2.2: Four Step Search procedure [3]. The motion vector is (3, -7).

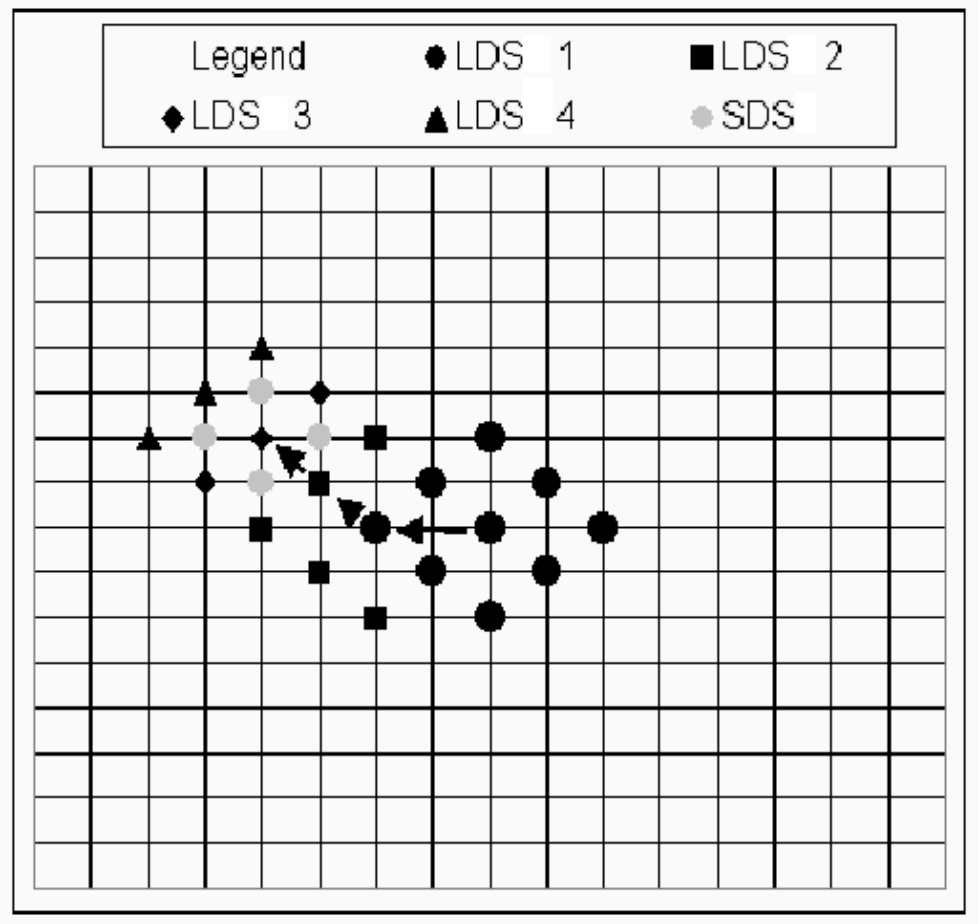


Figure 2.3: Diamond Search procedure. This figure shows the large diamond search pattern and the small diamond search pattern. It also shows an example path to motion vector (-4, -2) in five search steps-four times of LDS and one time of SDS [3].

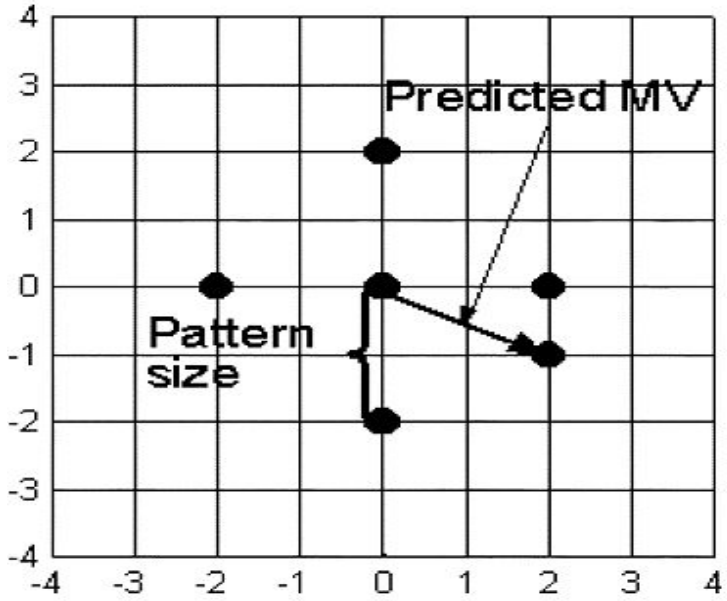


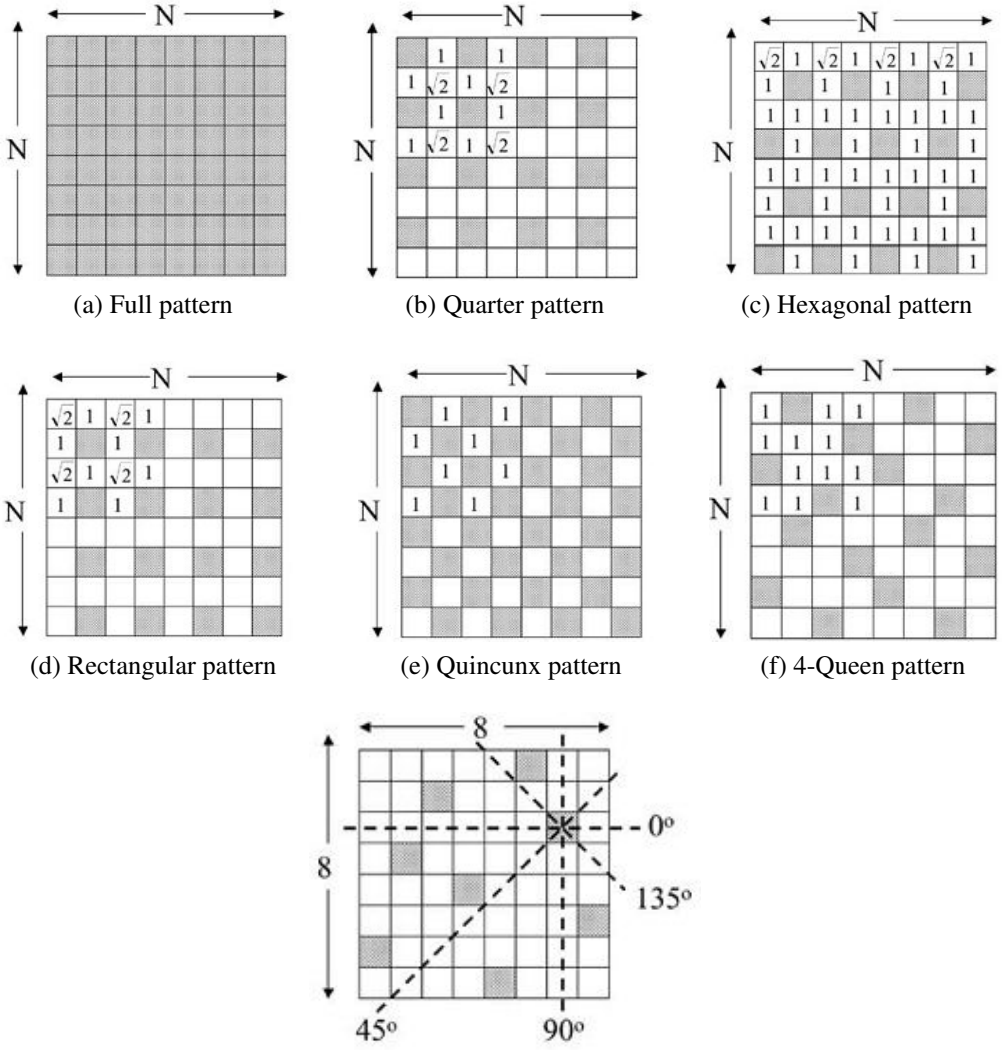
Figure 2.4: Adaptive rood pattern [43].

instead of the square shaped pattern as shown in Fig. 2.3. LDS is used in all the steps except for the last step in which the minimum distortion occurs at the center of the pattern, upon which SDS is used to obtain the MV. This approach significantly reduces the computational expense while achieving PSNR close to FSA. Adaptive Rood Pattern Search (ARP) [43] was proposed in order to overcome essentially two problems in DS. Firstly, in low motion content videos DS would oversearch by performing unnecessary LDS search while it can directly do the SDS. Secondly, in high motion content videos DS can be trapped in a local minima or it leads to a longer search path. ARP uses the fact that the MV of a macro-block can be effectively predicted from its neighboring macro-block (as shown in Fig. 2.4). Thus using the MV of the immediate left MB as the first step size, it carries on SDS until the center pixel is the minima. Some other algorithms in this class are [13, 17, 40, 42, 46, 50, 57, 58].

2.2 Fast Matching Techniques

In FSA, all the pixels in a macroblock are used to determine the distortion function when matching the current frame to the reference frame. Thus, for a macroblock of size 16×16, the SAD distortion is calculated for all the 256 pixels which leads to the high computational complexity of FSA. Thereby, reducing the number of pixels by selecting only a subsample of these samples can reduce computations significantly. Some examples for various pixel decimation patterns have been shown in Fig. 2.5.

In [4], a quarter subsampling pattern was used which reduced the computations by a factor of four. A hexagonal subsampling pattern [36] was used to overcome the drawback of aliasing effect without filtering. It is a popular belief that human visual system detects motion by processing edge signals [48]. Therefore, it makes sense to imitate human brain by using edge information but the main problem in such system arises when the video is quite uniform or when it is very noisy. In [10], a local pixel decimation scheme was proposed which divided the block into several regions and selecting more number of pixels from texture rich regions and it was subsequently improved in [8]. In [7], an edge oriented motion estimation was proposed. Hilbert scan was used in [53] to extract the global edge pixels in a one dimensional space. The algorithm does not divide the block into regions but it selects pixels only when they have important features. In [39], another fast matching method (FFSSG) based on pixel ordering was proposed. FFSSG attempts to sort the contributions to the gradient between the pixels to quickly discard invalid blocks. It proposes a more general framework for approaches used in [8] and [26]. In [26], the authors use Taylor's expansion to show the fact that the matching distortion at a certain position is proportional to the gradient magnitude of the reference block. The two algorithms are similar, but [26] uses a local area based gradient sorting, while FFSSG works on pixel based gradient sorting. Another pixel decimation approach using boundary-based (B4Q4) and genetic algorithm



(a) Full pattern (b) Quarter pattern (c) Hexagonal pattern

(d) Rectangular pattern (e) Quincunx pattern (f) 4-Queen pattern

(g) 8-Queen pattern

Figure 2.5: Various pixel decimation patterns [52].

based (BVG11) patterns was proposed in [49]. The authors speed up the process by matching the boundary regions only based on the observation that new objects usually enter macroblocks through their boundaries. They improve the boundary-based approach further, by performing a genetic-algorithm based search to find optimal set of pixels to be used for matching in order to have better spatial homogeneity and directional coverage.

Other algorithms such as pixel difference classification (PDC) [20], MiniMax criterion [12] were proposed in literature but it was observed that the N-Queen patterns [52] performed better than most in terms of reconstructed video quality and speed. In N-Queen pixel decimation approach, the spatial information of a $N \times N$ macroblock had been fully represented by the least number of pixels by selecting one pixel from each row, column and diagonal. Other important algorithms in this category includes but not limited to [9, 23].

2.3 Reduction in Bitwidth

Originally, eight-bit representation is used for each pixel but in [41] one-bit representation was used which yielded substantial reduction in arithmetic and hardware complexity with reduced power consumption. It transforms the current frame and the reference frame into one-bit representation using a transform kernel of 17×17 and use exclusive-OR (XOR) as the matching error criterion instead of SAD. This reduces the hardware area since a one-bit comparator is smaller than an eight-bit accumulator. In order to get rid of the binary multiplication complexity burden in 1BT, a multiplication-free one-bit transform (1MF) was introduced in [19]. Since a direct pixel truncation to one-bit was used, it resulted in significant performance loss. Thereupon, a modified one-bit transform (M1BT) and two-bit transform (2BT) were proposed in [18, 56] to improve degraded PSNR by adding a conditional search and an extra bit plane respectively. 2BT was subsequently improved by using positive and negative second derivatives in the derivation of second

bit plane in 2BT-SD [27]. A high performance systolic hardware architecture for 1BT based FSA was proposed in [2]. The hardware architecture performed full search ME for 4 MBs in parallel and they use less on-chip memory than previous 1BT by using a data reuse scheme and memory organization. In [6], another hardware implementation of 1MF was proposed. The authors used source pixel based linear array (SPBLA) hardware architecture for low bit depth ME which results in a genuine data flow scheme reducing the number of data reads from the current block, which in turn reduces the power consumption by at least 50% compared to conventional 1BT. Also, because of the binary nature of low bit-depth ME algorithms, their hardware architectures are more efficient than existing 8 bits/pixel representation based ME architectures. Some other algorithms in this class are [15, 31, 34, 37].

Each category of algorithms achieves different tradeoff between computational complexity and performance. The main drawback of approaches discussed in section 2.1 is that since the search window is minimized after the first iteration in general, it gets trapped in a local minima. Reduction in computational complexity obtained by using approaches in section 2.2 is also minimal as important pixels may be missed while evaluating the distortion function. Also, comparing their complexity directly is not easy as both run-time in software and area/bandwidth in hardware has to be compared. But, from the descriptions of these techniques it is evident that bitwidth reduction techniques is very fast in hardware implementation as fixed length pixel truncation produces a massive reduction in hardware complexity as one-bit comparator is smaller than an eight-bit accumulator; and in turn reduces power consumption. Also, with the one-bit/pixel or two-bit/pixel representation, it paves the way to make use of single instruction multiple data (SIMD) architecture with high degrees of parallelism in software. The above-mentioned observation lays the foundation for developing the Local Binary Pattern based strategy for motion estimation.

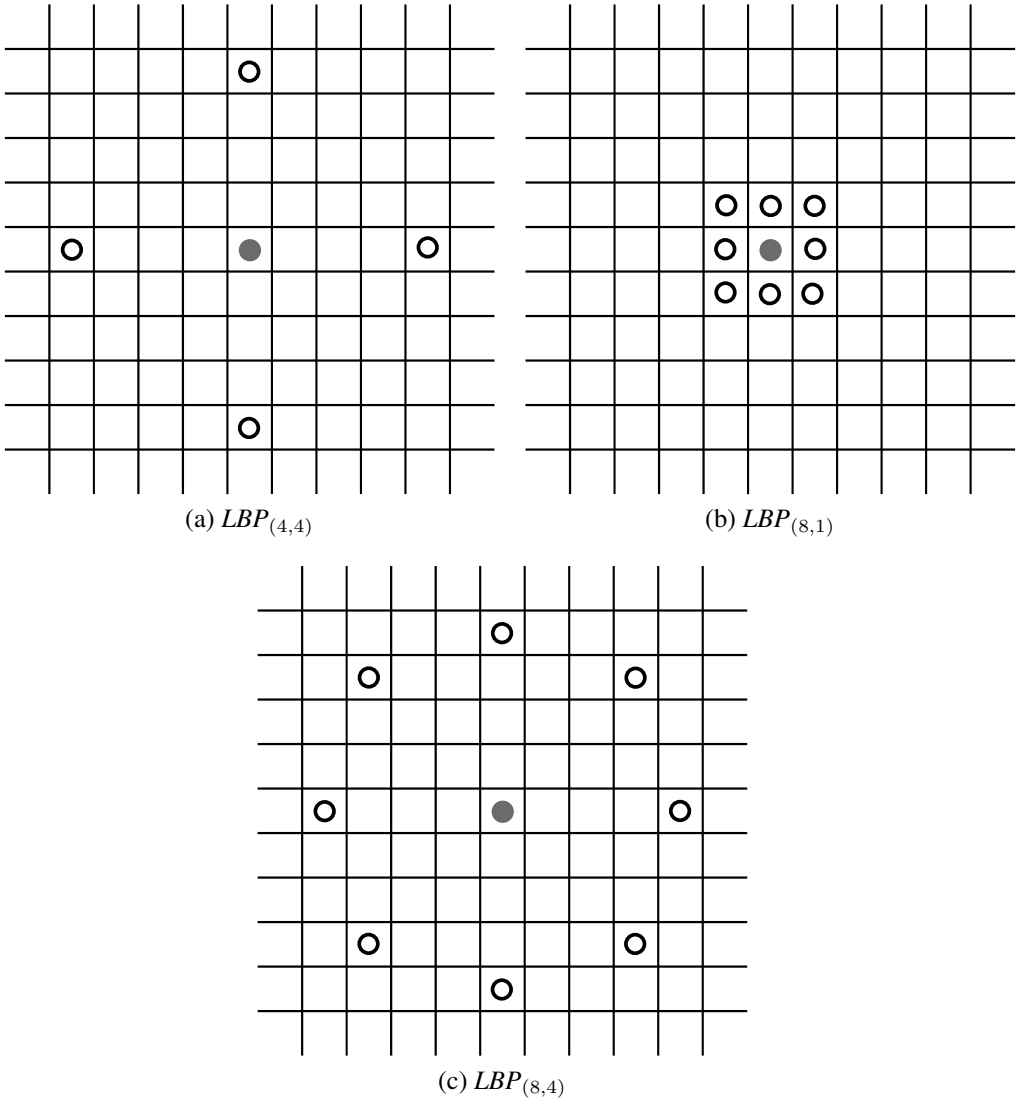
Chapter 3

Local Binary Patterns

Local binary pattern [38, 44] is a texture description operator with many features such as gray-scale invariance and no normalization in a neighborhood window which makes it favorable for use in motion estimation. The LBP operator T can be defined in a local neighborhood of a gray-scale image as the joint distribution of P image pixels and is given in (3.1),

$$T = t(g_c, g_0, \ldots, g_{P-1}). \tag{3.1}$$

where the center pixel intensity and its P neighbors are represented by g_c and $g_p(p = 0, \ldots, P-1)$. The P equally spaced pixels form a circle of radius R. The local binary patterns can be computed at circular neighborhoods of any quantization of the angular space and at any spatial resolution. Fig. 3.1 shows some examples of circularly symmetric neighbor sets for various (P, R) values, where P is the number of neighboring pixels and R is the radius. The joint difference distribution of the spatial characteristics can be found by,



(a) $LBP_{(4,4)}$ (b) $LBP_{(8,1)}$

(c) $LBP_{(8,4)}$

Figure 3.1: Local Binary Patterns

$$T \approx t(g_0 - g_c, g_1 - g_c, \ldots, g_{P-1} - g_c). \tag{3.2}$$

It is a highly discriminative operator which stores the various patterns in the neighborhood

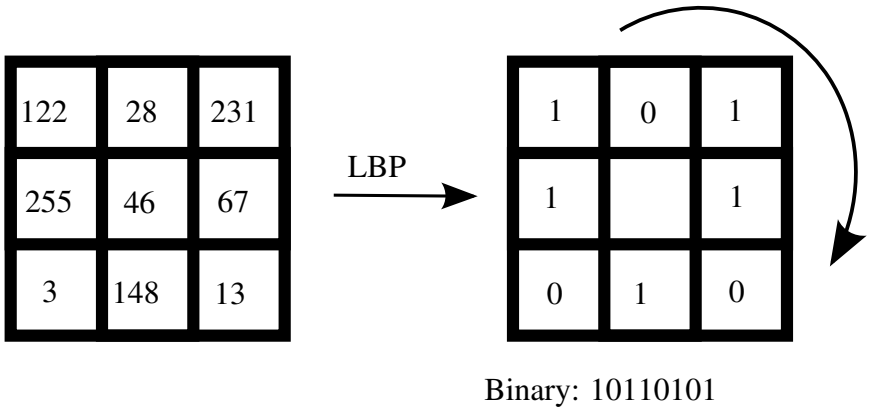


Figure 3.2: $LBP_{(8,1)}$ Operator

of each central pixel. For example, the differences are zero in a constant region, high along the direction of increasing or decreasing gradient, zero along the edge and very high in all directions for a dark/bright spot. In order to achieve invariance against scaling of gray scale, only the sign of the differences is considered ignoring the exact difference as shown in (3.3). Let $B(i)$ represent the binary bit at the neighboring pixel i, and let $I(c)$ represent the intensity at the center pixel c of the block, then the binary value at the i^{th} neighbor is given by,

$$B(i) = \begin{cases} 1, & if\ I(i) \geq I(c) \\ 0, & otherwise \end{cases} \tag{3.3}$$

The LBP operator $LBP_{(P,R)} = B(0, 1, ..., P-1)$ can be defined in a local neighborhood of a gray-scale image as the joint distribution of P image pixels at radius R. Thus, for each center pixel there is a corresponding $LBP_{(P,R)}$ value. This P bits pattern store valuable information which correspond to primitive structural micro-features on a local level, such as the presence of edges, corners and spots. This unique $LBP_{(P,R)}$ transform stores the characteristics spatial struc-

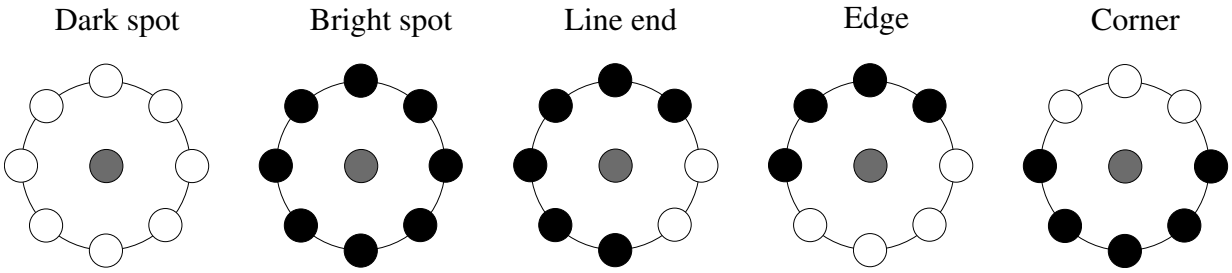


Figure 3.3: Various edge textures detected using $LBP_{(8,1)}$

ture of the local image texture around the center pixel g_c which has been exploited for developing the Local Binary Pattern based block motion estimation. LBP is a non-parametric method which means that no underlying assumptions are needed for computations. The transform is also direction invariant as it contains only the bit representation of the neighbor pixels without any encoding to decimal. It should be noted that no kernel is needed to calculate the comparison threshold, instead it is set to the center pixel intensity. For example, in Fig. 3.2 the LBP operator labels the pixels in a 3×3 neighborhood by thresholding each pixel value with the center value. Fig. 3.3 shows a few of the discriminative textures that can be efficiently identified using $LBP_{(8,1)}$ codes.

Chapter 4

Neighborhood Matching Approach

Various techniques to reduce computational complexity were discussed in Chapter 2. Among them, the bitwidth reduction techniques are quite robust to complex motion as well as their hardware and software implementation is quite simple. However, there are two main issues that needs to be addressed: 1) it has been inherently assumed that block estimation is invariant against scaling of gray-scale values of the pixels, and 2) there is no significant reduction in the number of operations when compared to full search techniques (FSA). Regarding the first issue, in most cases a lower bit representation has been used instead of eight bits which results in poor performance. Therefore, instead of scaling the whole image to a lower bit representation, some information about the scaling of the gray-scale values has been retained. By reducing the number of bit planes, the advantages of faster hardware implementation has been utilized but the number of operations has remained the same as FSA. Therefore, it would be beneficial to develop a faster algorithm that reduces the complexity further with acceptable performance, thereby enabling implementation on low capability mobile devices. The main objective of this work is to reduce the computations significantly so that it can be used on mobile devices. In the following sections,

the proposed approach has been discussed to effectively address the above-mentioned issues.

4.1 Full Search Local Binary Patterns (FS-LBP)

Local binary pattern (LBP) is a texture description operator with many features such as gray-scale invariance and no normalization in a neighborhood window which makes it favorable for use in motion estimation. LBP is a non-parametric method which means that no underlying assumptions are needed for computation. LBP transform is also directional invariant which is achieved by comparing each of the neighbors with the central pixel as mentioned in the previous section. The transform is also direction invariant as it contains only the bit representation of the neighbor pixels without any encoding to decimal, thus it is independent of directional encoding. It should be noted that no kernel is used to calculate the comparison threshold as used in [18, 19, 27, 41], instead it is set equal to the center pixel intensity. Thus, for each pixel there are P bits of information about its P neighbors. These P bits pattern store valuable information which correspond to primitive structural micro-features on a local level, such as the presence of edges, corners and spots; hence, they can be used to describe the region around a center pixel efficiently. The objective is to use these local texture features to develop a new matching criterion which not only captures the gray-scale variations between macro-blocks but also the spatial texture of that local region.

In order to derive a motion vector with FS-LBP representation, a combination of the sum of absolute difference (SAD) and the number of non-matching neighbors (NNMN) is used as the matching criterion as SAD maintains the original pixel intensity and NNMN contains the neighborhood information. The SAD of two blocks of size $N \times N$ is given by,

$$\text{SAD}(m,n) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} |I^t(i,j) - I^{t-1}(i+m, j+n)| \tag{4.1}$$

where I^t and I^{t-1} are the intensity values of the current and the reference image frames, and (m,n) denotes the candidate displacement. The NNMN is given by,

$$\begin{aligned} \text{NNMN}(m,n) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \textit{LBP}^t_{(P,R)}(i,j) \otimes \textit{LBP}^{t-1}_{(P,R)}(i+m, j+n); \\ -s \leq m,n \leq s-1 \end{aligned} \tag{4.2}$$

where $\textit{LBP}^t_{(P,R)}$ and $\textit{LBP}^{t-1}_{(P,R)}$ are the LBP transforms for the current and the reference frames respectively, and s is the search range. NNMN captures the number of mismatching neighbors around the central pixel of a macro-block in the current frame and the reference frame. The cost function is defined as,

$$\text{Cost}(m,n) = \alpha * \text{NNMN}(m,n) + (1-\alpha) * \text{SAD}(m,n) \tag{4.3}$$

where α is the scaling factor. The motion vector (MV) is given by,

$$\begin{aligned} \text{MV} = (u,v) \mid \text{Cost}(u,v) \leq \text{Cost}(m,n); \\ -s \leq m,n \leq s-1 \end{aligned} \tag{4.4}$$

The cost function incorporates the advantage of SAD as well as it gives lesser error if most of the neighbors are matching, but at the cost of added computations.

4.2 Reduced Complexity Local Binary Patterns (RC-LBP)

The computational complexity of the proposed FS-LBP technique is higher than FSA as it has to compute the additional local binary patterns and also uses a new cost function instead of classical SAD. Thus in order to overcome the high computational complexity of the LBP technique, a reduced complexity LBP (RC-LBP) is proposed in this section. As observed in LBP, a small region around a central pixel can be effectively represented by an $LBP_{(P,R)}$ code that contains useful information about the structural micro-features. The objective is to find a good approximation of that region (a sub-block) as a whole in terms of LBP rather than the usual gray-scale values for each pixel. Thus, a sub-block (SB) can be effectively represented by P single bits of information instead of 8-bits for each pixel covered by a sub-block in the macro-block. Let S denote the sub-sampling ratio of MB to SB. For example, in Fig. 4.1 it can be seen that each macro-block of size 16×16 pixels is divided into sub-blocks of size 4×4.

To find an accurate motion vector, a two step search process has been employed in order to accomplish a fast motion estimation. The first search works on the sub-sampled LBP image instead of the full image so as to find a good starting point for the second step which involves local refinement thereby, avoiding unnecessary full search and reduce the risk of being trapped in a local minimum. The reduced complexity algorithm performs the following steps for each macro-block (MB):

Step 1 (LBP Transform): Compute the $LBP_{(P,R)}$ transform for only the pixels shown in Fig. 4.1 for a MB in the current and the reference frame. The LBP transform for each selected pixel uses the same values of P and R. A good estimate of the general motion of a macro-block can be obtained by using the sub-block (SB) since block motion estimation is based on the assumption that all pixels in a macro-block move by the same amount. Note that for sub-sampling ratio S each 16×16 MB in the original image is represented by $(16/S)\times(16/S)$ SB's

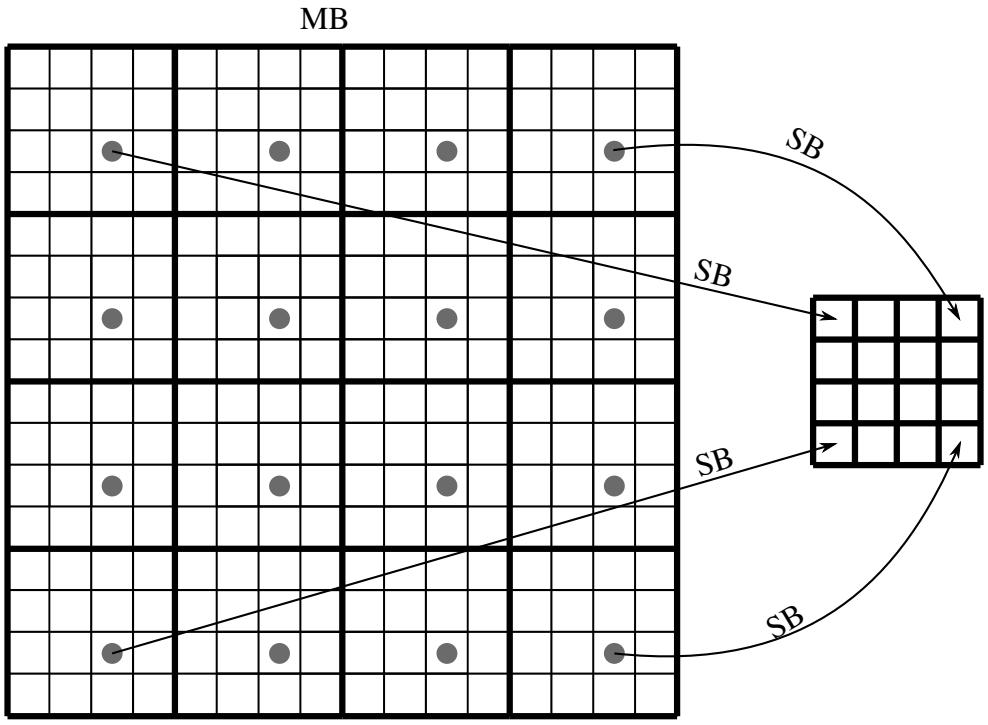


Figure 4.1: Macro-block division into Sub-block for $S = 4$

in the transformed image which store the LBP values. Thus, instead of computing the LBP transform for each of the 256 pixel in a MB, the image is analyzed in a non-overlapping window of $S \times S$ pixels in raster order. Thereby, effectively dealing with a subsampled image which has been subsampled by S in each of the two directions.

Step 2 (Cost Function): Compute the cost function according to (4.3). Thus, effectively giving the MV for each SB in a MB. It should be noted that now the NNMN is calculated for only one center pixel and SAD is just the intensity difference of the center pixel instead of all the pixels as was done in the FS-LBP method. The MV's are calculated for each SB (which is equivalent to $S \times S$ block in the original image) with each $\pm$ 1 movement in the transformed image equivalent to $\pm S$ in the original image. Thus, instead of calculating the MV's in steps

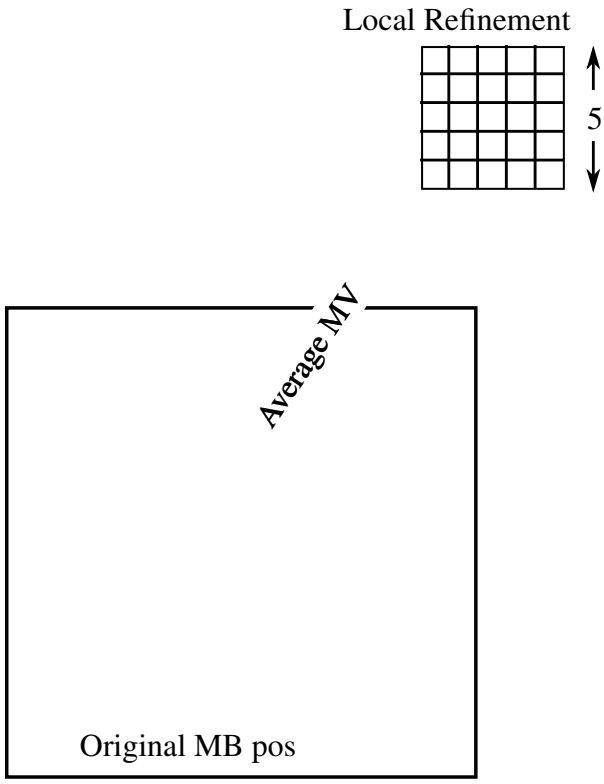


Figure 4.2: Local refinement of average MV in a window of ± 2

of 1 pixel, the movement is quantized in steps of S. Previously, the search range was ± 16 but now the search range is reduced to $\pm(16/S)$ representing the same amount of motion. This step reduces the required number of operations drastically. The main idea is to divide and conquer the size of a MB and work with smaller SB.

Step 3 (Average MV): The individual MV of each SB is combined into an average motion vector which represents the general sense of motion of the macro-block. It is also inherently assumed in BMA's with MB size of 16×16 that a body is rigid and the block represents the translation motion of the entire body. Thus, their performance is poor when objects within the same MB move in different directions or with different velocities. Therefore, it is more likely that these objects moving in different directions will be found in smaller SB's with the afore-

mentioned assumption being valid. Consequently, the quality of motion vectors improves with the use of smaller blocks, but it increases the bit rate. Thus, a tradeoff between block size and bit rate exists. In order to take advantage of smaller block size by maintaining the bit rate, average motion vectors are used to represent them.

Step 4 (SDS Refinement): Average MV gives the general motion of the entire MB which is subsequently refined locally. The average MV identified is hopefully as close to the global minimum as possible. Both FSA and small diamond search pattern (SDS) [59] has been investigated in our experiments for local MV refinement and the results show that SDS reduces the search points by 45% to 80% while maintaining similar PSNR. Therefore, SDS has been used for refinement procedure as shown in Fig. 4.2. It is observed in our experiments that SDS refinement in a small, compact, and fixed-size window of ± 2 around the average motion vector yields acceptable performance with minimum searches. Thus, average MV helps to reach the local region of global minimum distortion directly and SDS refinement leads to the optimal MV efficiently.

4.3 Simulation Results and Analysis

The most widely used and simplest full-reference quality metric is the peak signal-to-noise ratio (PSNR). It is simple to calculate with clear physical meaning and mathematically convenient to study as an optimization criteria. But, it does not give a very good match to the perceived visual quality. It has been shown in [54, 55], that structural similarity (SSIM) metric provides more resemblance to the human visual system as it compares the local patterns of pixel intensities which have been normalized for contrast and luminance instead of intensity difference error-pooling. This section aims to evaluate the effectiveness of the proposed motion estimation based on LBP.

4.3.1 Parameters P, R, α and S

As shown in Fig. 3.1, there are many possible combinations for various values of the number of neighboring pixels (P) and the radius of operation (R). Extensive experiments were performed with different values of (P, R) such as $\{(4,1),(4,2),(8,1),(8,2),(8,4),(16,4)\}$. Our experimental results show that these yield almost similar performance in terms of PSNR (with difference within 0.05 dB). $(P = 8, R = 4)$ has been chosen so that the LBP value represents the texture of a middle range value instead of being too local or too global. Thus, it effectively captures the neighborhood information in a window of size 9×9.

The parameter α used in (4.3) is used to determine which of the two: NNMN or SAD, becomes the main contributor to the final cost distortion function. Variable $\alpha = 0$ is equivalent to using only SAD and $\alpha = 1$ denotes using NNMN as the only deciding criterion in the distortion function. The value of α varies from frame to frame as well as for different sequences. By applying FS-LBP on 8 typical video sequences, it was found that the mean NNMN per MB is 3.8075 and mean SAD per MB is 25.8622, therefore the best reconstruction quality performance was obtained for $\alpha = 0.9$. Therefore, for cost calculations with almost similar intensity value pixels, NNMN becomes the deciding distortion criterion but when there is large variation in pixel intensities, SAD becomes the major contributor to the cost distortion function. Therefore, this type of cost function not only eliminates the complete dependence on the pixel intensities but also takes into account the spatial structure of a local region. Thus, $\alpha = 0.9$ confirms with the theory and is validated by experiments. But (4.3) contains two multiplications which is computationally expensive compared to additions, therefore a multiplication-free cost functions as defined in (4.5) is used which gives almost similar performance with β being the new scaling factor given by (4.6).

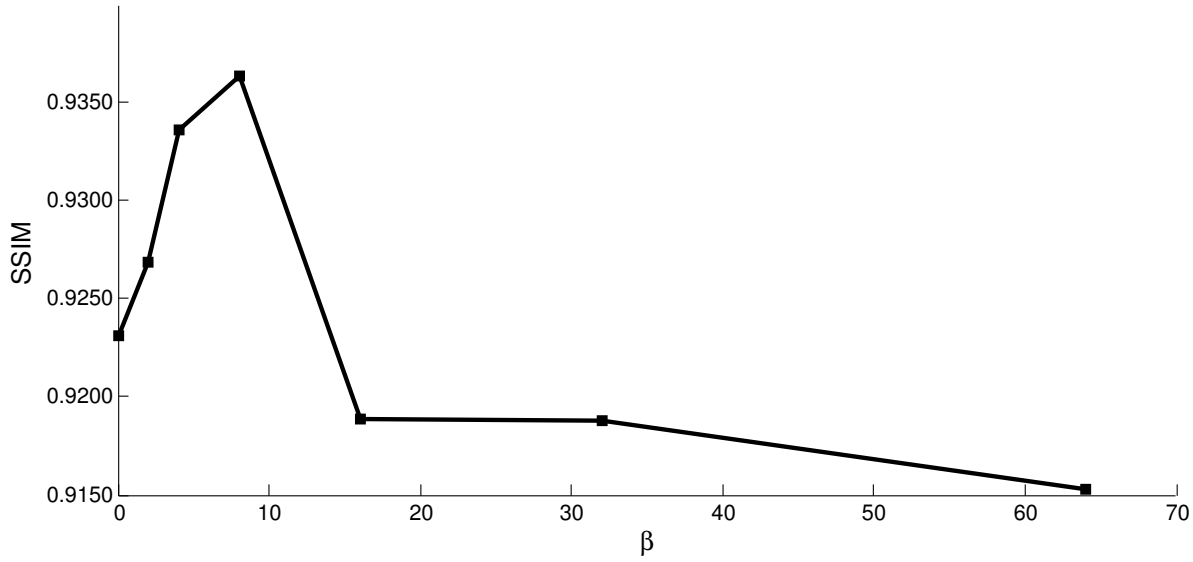


Figure 4.3: Variation of SSIM performance with β for Foreman image sequence.

$$\text{MFCost}(m,n) = \beta * \text{NNMN}(m,n) + \text{SAD}(m,n) \tag{4.5}$$

$$\beta = \frac{\alpha}{1-\alpha} \tag{4.6}$$

Fig. 4.3 shows the performance of FS-LBP in terms of SSIM of the reconstructed frames for Foreman image sequence which essentially contains almost all types of motion including camera panning, fast and slow motion. It can be seen that using LBP features gives better performance than just using pixel intensities (for β = 0). Also, if only LBP features are used neglecting the pixel intensities (for β = 64), the performance degrades. Therefore, the value of β is set to 8 which is closest to $\alpha = 0.9$ which gives the value of β to be 9. Also, it can be easily achieved in hardware by a bit shift operation without using multiplication.

RC-LBP was tested on the test sequences with varying values of S such as $\{2, 4, 8\}$. Since lower value of S would result in higher complexity and higher S would result in performance loss, it was conservatively chosen as 4, which achieves fairly good computational gain without causing noticeable degradation on visual quality.

4.3.2 SSIM and PSNR performance

FS-LBP and RC-LBP are compared with FSA, 1BT, 1BT-MF, 2BT and 2BT-SD. These algorithms employ exhaustive full search algorithm for fair comparison. Eight sequences Foreman, Football, Tennis, Coastguard, Mother & Daughter, Salesman, Miss America (CIF 352×288) and Akiyo (QCIF 176×144) has been used as test sequences with a macro-block size of 16×16 and a search range of ±16 pixels. PSNR and SSIM values for the video sequences with image frames reconstructed from the previous frames are utilized for statistical evaluation of motion estimation accuracy. These reconstructed frames are furthermore used for visual evaluation of motion estimation accuracy.

Table 4.1: Performance comparison in SSIM

Sequence	**FSA**	**FS-LBP**	**1BT**	**1BT-MF**	**2BT**	**2BT-SD**	**RC-LBP**
Foreman	0.9231	0.9364	0.9069	0.9091	0.8999	0.9059	0.8379
Football	0.7779	0.7838	0.7574	0.7612	0.7586	0.7628	0.6851
Tennis	0.8964	0.8993	0.8896	0.8905	0.8897	0.8893	0.7862
Coastguard	0.9145	0.9153	0.9092	0.9102	0.9075	0.9103	0.8214
Akiyo	0.9943	0.9947	0.9941	0.9942	0.9933	0.9930	0.9813
Mother	0.9699	0.9703	0.9570	0.9548	0.9577	0.9633	0.9470
Salesman	0.9413	0.9419	0.9322	0.9351	0.9356	0.9351	0.9212
Miss America	0.9243	0.9255	0.9090	0.9089	0.9113	0.9131	0.8939
Average	0.9177	0.9209	0.9073	0.9080	0.9067	0.9091	0.8593

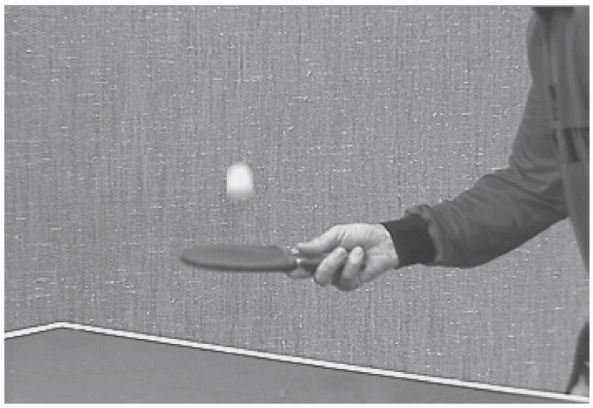
The SSIM and PSNR evaluation for various sequences are shown in Table 4.1 and Table 4.2.

Table 4.2: Performance comparison in PSNR (dB) and average number of SDS search positions per MB for RC-LBP

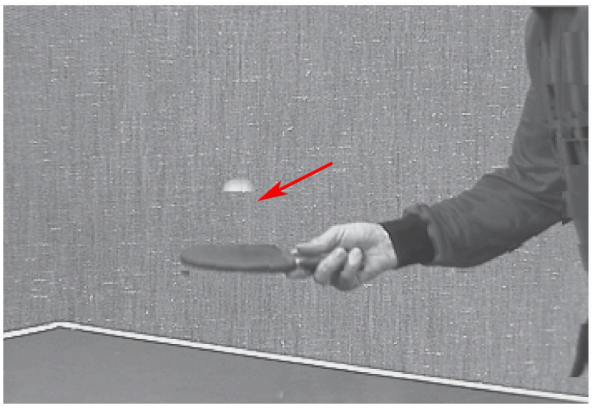
Sequence	**FSA**	**FS-LBP**	**1BT**	**1BT-MF**	**2BT**	**2BT-SD**	**RC-LBP**
Foreman	32.12	32.03	30.47	30.39	30.78	31.16	27.86 (8.2)
Football	23.68	23.26	22.38	22.31	22.72	22.98	21.17 (8.2)
Tennis	29.22	28.89	28.20	28.19	28.44	28.83	23.96 (8.1)
Coastguard	30.48	30.47	29.88	29.87	29.88	30.20	25.49 (8.5)
Akiyo	43.56	43.51	43.39	43.45	43.45	43.47	42.10 (6.0)
Mother	40.47	40.39	37.06	36.22	37.79	39.30	37.54 (7.9)
Salesman	35.34	35.17	31.72	32.95	34.91	35.08	33.38 (7.8)
Miss America	37.77	37.59	35.75	35.53	36.77	36.87	34.04 (8.1)
Average	34.08	33.91	32.36	32.36	33.09	33.49	30.69 (7.8)

Table 4.1 shows that when the different sequences are evaluated using SSIM metric, FS-LBP outperforms FSA. FS-LBP performs better than FSA in terms of structural similarity which produces a smooth video perception. FS-LBP produces a better subjective viewing quality because it takes into account the joint distribution of neighborhood pixels. Since 1BT, 1BT-MF, 2BT and 2BT-SD evaluate the motion vectors using a lower bit representation of the central pixel, their performance is worse than FSA. SSIM performance of RC-LBP is lower, but it is computationally very efficient as compared to other methods. Table 4.2 shows that the PSNR performance of RC-LBP is lower. However, the performance of RC-LBP can be acceptable for limited computational capability devices, such as smartphones. The main reason for performance degradation of RC-LBP is that it takes into account the neighborhood of the central pixel and if the number of non-matching neighbors is high then its cost function is high. Thus, FS-LBP and RC-LBP preserves the structure around a central pixel rather than fitting the macro-block from reference frame to current frame in order to minimize SAD. Since, RC-LBP is a reduced complexity approach its performance is worse than FS-LBP.

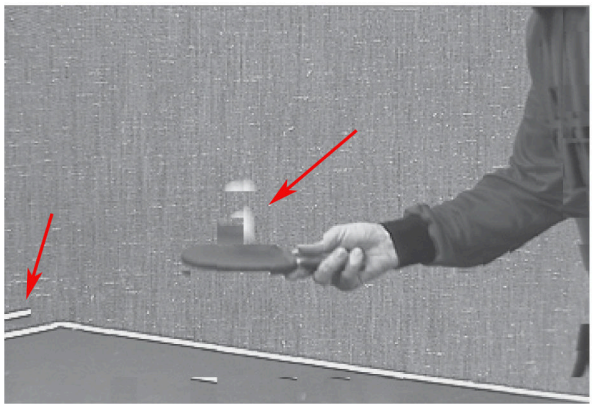
From Table 4.1 and Table 4.2, it can be seen that the performance of FS-LBP is worse than



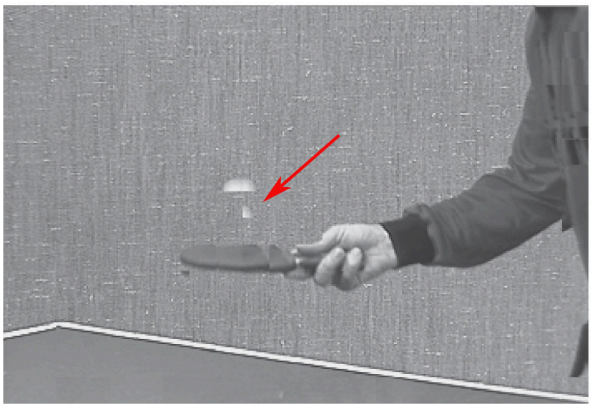
(a) Original frame



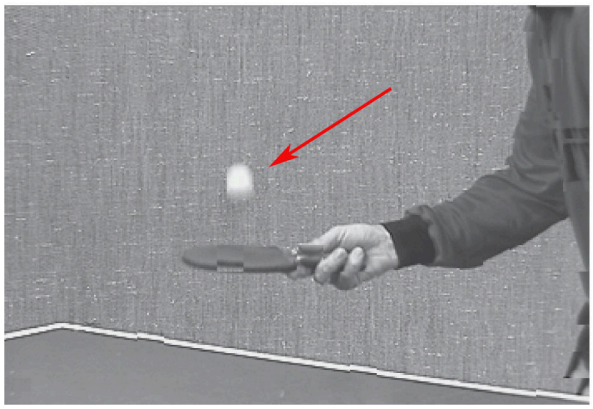
(b) FSA (PSNR: 27.60dB, SSIM: 0.7647)



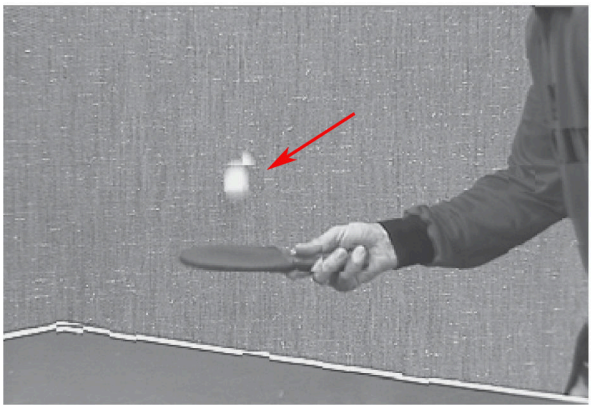
(c) 1BT (PSNR: 26.12dB, SSIM: 0.7498)



(d) 2BT-SD (PSNR: 26.98dB, SSIM: 0.7563)

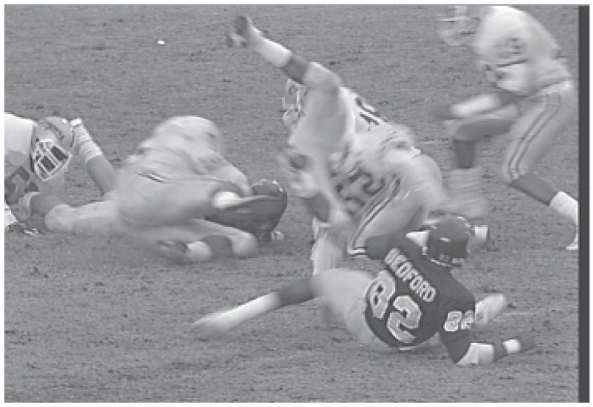


(e) FS-LBP (PSNR: 27.28dB, SSIM: 0.7675)

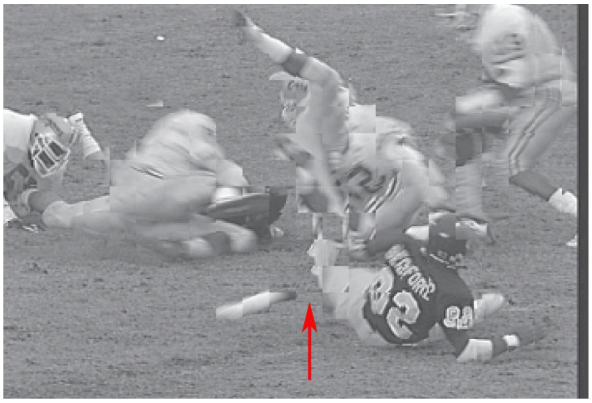


(f) RC-LBP (PSNR: 23.04dB, SSIM: 0.6574)

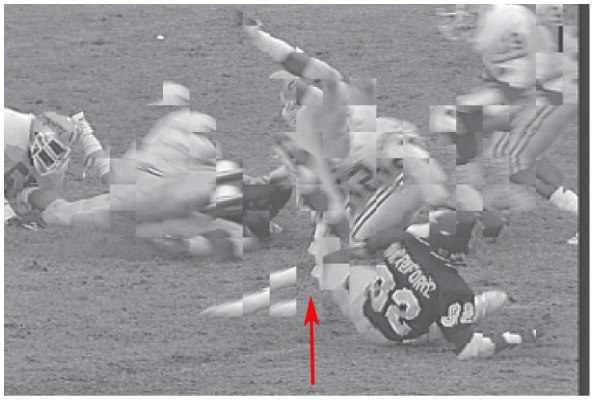
Figure 4.4: Sample reconstructed frame #28 from previous frame in Tennis sequence.



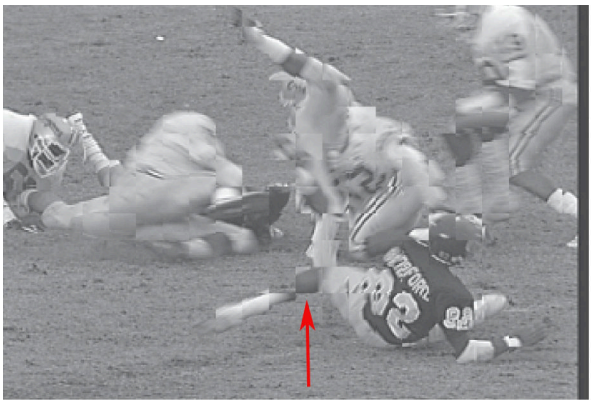
(a) Original frame



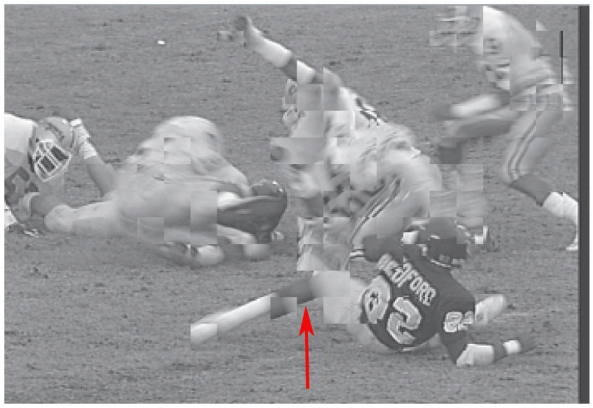
(b) FSA (PSNR: 24.18dB, SSIM: 0.7548)



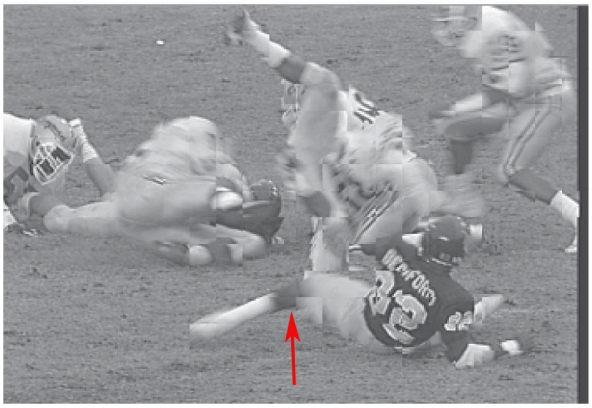
(c) 1BT (PSNR: 22.11dB, SSIM: 0.7270)



(d) 2BT-SD (PSNR: 23.46dB, SSIM: 0.7398)



(e) FS-LBP (PSNR: 24.07dB, SSIM: 0.7643)



(f) RC-LBP (PSNR: 21.67dB, SSIM: 0.6729)

Figure 4.5: Sample reconstructed frame #15 from previous frame in Football sequence.

(a) Original frame

(b) FSA (PSNR: 33.63dB, SSIM: 0.9342)

(c) 1BT (PSNR: 31.86dB, SSIM: 0.9200)

(d) 2BT-SD (PSNR: 32.59dB, SSIM:0.9193)

(e) FS-LBP (PSNR: 33.57dB, SSIM: 0.9370)

(f) RC-LBP (PSNR: 29.84dB, SSIM: 0.8813)

Figure 4.6: Sample reconstructed frame #10 from previous frame in Foreman sequence.

FSA in terms of PSNR quality metric but it is better than FSA in terms of SSIM quality metric. Therefore, it is important to study the performance of these techniques by visual inspection of the reconstructed frames. Subjective visual quality of the reconstructed frames shows that the proposed FS-LBP and RC-LBP give visually more appropriate frames compared to FSA, 1BT and 2BT-SD. For Tennis sequence in Fig. 4.4, it can be seen that even the FSA is not able to reconstruct the full frame accurately as the ping-pong ball is half missing, since FSA is implemented using only the pixel intensity values while FS-LBP produces a better visual quality frame. 1BT results in bad motion vector and 2BT-SD produces better visual quality than FSA. Although there is some distortion in the reconstruction of the racket in FS-LBP and the edge of the table in RC-LBP but they produce improved quality reconstructed frames over other techniques. In Fig. 4.5 for Football sequence, it can be noticed that the visual appearance of RC-LBP outperforms both FSA and FS-LBP. FSA is not able to reproduce the leg of the player accurately and 1BT results in substantial amount of bad motion vectors. 2BT-SD produces better visual results than FSA but RC-LBP clearly outperforms other techniques. It performs better than FS-LBP as it evaluates the motion vector in SB's of 4×4 rather than 16×16. Fig. 4.6 shows sample results for Foreman sequence. It can be seen that FSA, 1BT, 2BT-SD and FS-LBP results in degraded visual appearance as they produce bad motion vector for the mouth region. RC-LBP outperforms even FSA in terms of visual quality and reconstruction of the mouth region. RC-LBP results in better results since it employs smaller SB as mentioned earlier. Therefore, RC-LBP can be used with recent video coding standards like MPEG-4 that supports smaller block sizes for motion estimation.

The main reason for the better visual quality of the reconstructed frames using FS-LBP and RC-LBP is that they take advantage of the correlation between a center pixel and its neighbors by representing it in a compact LBP code. The matching distortion evaluation is done on this LBP code instead of the individual pixels, therefore pixels belonging to high textured regions like

edges and corners get grouped together into a code, and therefore processed together leading to better reconstruction of the object as a whole. The reason for the quality degradation for RC-LBP in terms of PSNR and SSIM metric is that it focusses more on producing a better reconstructed frame rather than minimizing each pixel to pixel distortion error, therefore suffering more when the background consist of low texture or a noisy region. Flat or background regions of an image usually contain less edges, therefore, the LBP approach is not able to accurately distinguish them and hence, the drop in reconstructed frame's SSIM quality. One possible solution is to avoid using LBP motion vector search operations for these macroblocks, and deriving the motion vectors by prediction from neighboring blocks. Although some adverse motion vectors are also encountered for RC-LBP approach compared to FS-LBP results, the amount of bad motion vectors and their visual impact are significantly lower than that of 1BT and 2BT-SD.

4.3.3 Complexity analysis

In order to measure the computational complexity, the analysis on the complexity is presented in Table 4.3 and 4.4. For FS-LBP, eight comparison operations are required per pixel for transforming the image to LBP. For motion estimation, each 16×16 MB requires same addition, comparison and absolute operations as FSA but it requires an additional overhead in terms of shift and boolean operations to calculate the NNMN.

Table 4.3: The number of operations per pixel required for transforming each image

Operation	**FS-LBP**	**1BT**	**1BT-MF**	**2BT**	**2BT-SD**	**RC-LBP**
Addition	0	25	16	2.84	10.01	0
Multiplication	0	1	0	1.06	0.05	0
Comparison	8	1	1	3	2	0.5
Shift	0	0	1	0	1	0
Boolean	0	0	0	1	0	0

Table 4.4: Computational complexity for matching in number of operations per 16x16 MB with a 16 pixel search range

Operation	FSA	FS-LBP	1BT	1BT-MF	2BT	2BT-SD	RC-LBP
Addition	262144	262144	262144	262144	262144	262144	4100
Comparison	1024	1024	1024	1024	1024	1024	1032
Shift	0	262144	0	0	0	0	17
Boolean	0	2097152	262144	262144	786432	1048576	8192
Absolute	262144	262144	0	0	0	0	3072
Total	525312	2884608	525312	525312	1049600	1311744	16413

For RC-LBP, the number of comparison operations required to transform the image is half per pixel as it is evaluated for only one pixel in a 4×4 block. Since the average number of SDS search positions for refining the average MV for various test sequences is 7.8 (~ 8) as in Table 4.2, it has been used for calculating the computation cost. To calculate the number of operations in motion estimation, consider a 16×16 macro-block (MB) which has sixteen sub blocks (SB). It is to be noted that initially MV is calculated for each SB and then it is averaged to get the average motion vector for each 16×16 MB. For each SB, 128 additions are required as the cost function has 2 additions. Thus total number of additions for each MB is sixteen times that for each SB, additional sixteen for computing the average and 2048 additions for SDS refinement. Similarly, the number of comparison required for each MB is sixteen times that of a SB $(= 64)$ and eight more for refinement stage. Shift operation is required to compute the MFCost function multiplication for each SB $(= 64)$ and the average MV for each MB $(= 1)$. For each SB, boolean operation $(= 512)$ is required to compute the NNMN. Sixty four number of absolute operation required for each SB and 2048 for SDS stage.

It can be seen that the computational complexity of RC-LBP is very low when compared to the other techniques while maintaining good visual performance. RC-LBP is the fastest algorithm, with 96.88% improvement in computational complexity over 1BT and 98.75% improve-

ment over 2BT-SD. Another advantage is its computational simplicity as the LBP operator can be realized with a few operations in a small neighborhood and a lookup table. The objective was to reduce complexity by representing multiple pixel values with a single LBP feature. Owing to high discrimination performance in searching for the best MV, it is able to produce near-optimum MV's with only a fraction of the computational burden of that of the other methods. A limitation of RC-LBP is that it is not able to find accurate motion vectors for macroblocks belonging flat/background or unfocussed regions, so it will work best when the video is focussed sharply. FS-LBP results in considerable improved visual quality frames and RC-LBP is a computationally efficient technique which provides a reasonable motion estimation performance along with its potential application in recent video compression schemes where smaller block sizes are employed.

Chapter 5

Edge Detection Approach

5.1 Partial Distortion Elimination based on LBP (PLBP)

The partial distortion elimination (PDE) algorithm is a common lossless fast matching algorithm that eliminates redundant calculations during the computation of SAD within a block. Classical BMAs' compare the final SAD of the candidate search position to the current minimum SAD in order to determine the best motion vector. But, PDE uses the partial sum of differences by stopping midway to eliminate impossible candidates before finishing the full SAD calculation. The row-based partial SAD (pSAD) till the k^{th} row of a block of size $N \times N$ is given by

$$\begin{aligned} \text{pSAD}_k(u,v) &= \sum_{i=0}^{k-1} \sum_{j=0}^{N-1} |I^t(i,j) - I^{t-1}(i+u, j+v)|; \\ & 0 \leq k \leq N-1 \end{aligned} \tag{5.1}$$

The partial SAD is computed until it becomes equal to or greater than the minimum SAD

already found. The matching is performed row wise and the check is performed after each iteration of k until it satisfies the condition for $k \leq N-1$, thereupon the matching is terminated and the candidate search position is rejected. Thus, the order in which the searching is done hugely impacts the matching phase. For example, if a good motion vector prediction is found early then the matching phase gets tighter distortion bounds thereby skipping lots of unnecessary computations.

The order in which the pixels within a block gets matched to compute the partial SAD also affects the speed of PDE. If the highest contributions to SAD pixels can be matched early then the faster the pSAD will satisfy the rejection condition. Therefore, instead of using a fixed row wise order of matching pixels, a generic order T can be used, where T contains N^2 elements each corresponding to a pixel in a reference $N \times N$ block in some order.

SpiralPDE [1] is considered to be the simplest approach. Its searching strategy is based on a spiral from the center of the search window thereby, maximizing the chance of an initial good minimum SAD prediction. Its matching is done in a top-to-bottom raster scan order. It has been inherently assumed in SpiralPDE that the highest contribution pixels would be found in the first row of a block. This assumption is not always valid and therefore, a generic order can be used which tries to address this issue more effectively by studying the relationship between a pixel and its surrounding neighbors in a block.

In [39], the authors provided the Taylor series expansion of the pixel distortion. Let (i, j) be the position of a pixel in a block, the pixel distortion between the current block and the candidate block with the position (u, v) can be given by,

$$D_{u,v}^{i,j} = |I^t(i,j) - I^{t-1}(i+u, j+v)| \tag{5.2}$$

where $I^t(i,j)$ denotes the pixel intensity at pixel position (i,j) within the current block and

$I^{t-1}(i+u, j+v)$ denotes the pixel intensity at pixel position $(i+u, j+v)$ within the candidate block located at the position given by its motion vector (u, v). Therefore, if the value of $D^{i,j}_{u,v}$ is known for the motion vector $(0, 0)$, then the neighboring values can be approximated using the Taylor series expansion given by,

$$D^{i,j}_{u,v} \simeq D^{i,j}_{0,0} + \nabla_{u,v} D^{i,j}_{0,0} \cdot ((u, v) - (0, 0)) \tag{5.3}$$

According to (5.3), the value of $D^{i,j}_{u,v}$ is approximated from the sum of $D^{i,j}_{0,0}$ and the differential term obtained through the inner product of the gradient vector of $D^{i,j}_{0,0}$ and the position difference vector. A simple way to approximate the pixel distortion given by (5.3) would be to use only the first term that is the distribution of differences between the current block and the candidate block at the center of the search window and use these differences to sort the positions in a decreasing order to obtain the sequence for SpiralPDE matching. But the problem arises when the block being coded is quite similar to the area at the center of the search window (in case of low motion content videos), the first term of the Taylor series are quite small and almost always equal to zero which would render sorting meaningless. Therefore, an optimized matching order cannot be obtained by only using the first term. The spatial gradient term also has to be considered but, its main drawback is very high computational complexity of calculating the magnitude of the gradient for each pixel in a block. Thus, the main motivation of the proposed PLBP approach is based on this idea of computing the gradient in an efficient as well as an effective manner so as to get an optimized matching order.

The idea is to reduce the second term in (5.3) to a simpler term by using the LBP approach. As mentioned earlier, in LBP, a small region around a central pixel can be effectively represented by an $\boldsymbol{LBP}_{(P,R)}$ code that contains useful information about the neighboring pixels. Let us formally define the difference LBP ($\boldsymbol{LBP}^{D}_{(P,R)}$) as the LBP patterns consisting of the count of the number

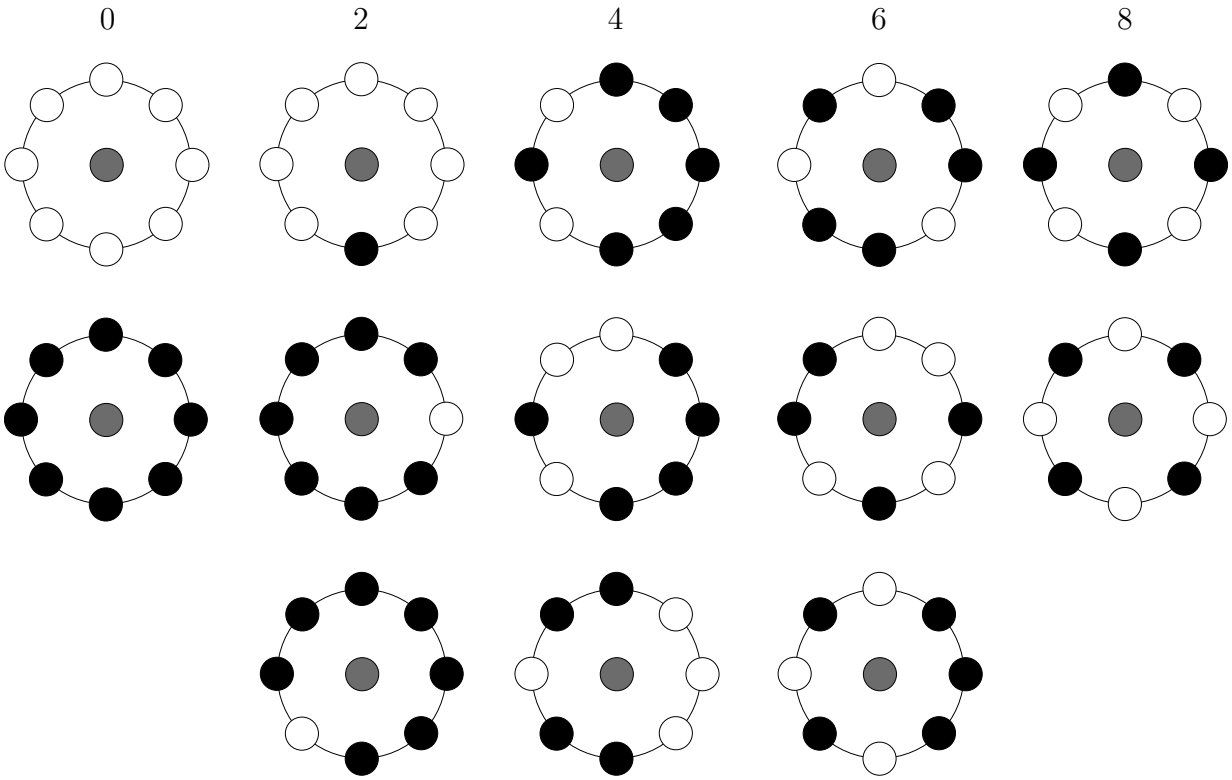


Figure 5.1: Various possible $LBP^{D}_{(8,R)}$ codes from basic LBP patterns

of binary-bit transitions from either one-to-zero or zero-to-one in a cyclic order.

$$LBP^{D}_{(P,R)} = \sum_{p=0}^{P-1} |g_p - g_{p+1}| \tag{5.4}$$

The $LBP^{D}_{(P,R)}$ operator essentially captures the non-uniformity around a central pixel. It should be noted that there are a total of five possible patterns for $P = 8$, these are $\{0, 2, 4, 6, 8\}$. Fig. 5.1 shows some of the LBP codes that generate these difference patterns. Considering (5.3), we approximate the gradient of distortion $D^{i,j}_{0,0}$ using these difference LBP patterns. Therefore, by evaluating $LBP^{D}_{(P,R)}$ for each pixel in a block and sorting the positions in a decreasing order, an optimized matching order can be obtained. In practice, $LBP^{D}_{(P,R)}$ is best implemented by

using a direct mapping look-up table that converts the basic LBP patterns into their difference LBP correspondents. PLBP is determined only once in the current block, and then this order can be used for all the candidate blocks within the search window. The proposed PLBP algorithm performs the following steps for all the pixels in the current block:

Step 1: Compute the $LBP_{(P,R)}$ patterns for each pixel.

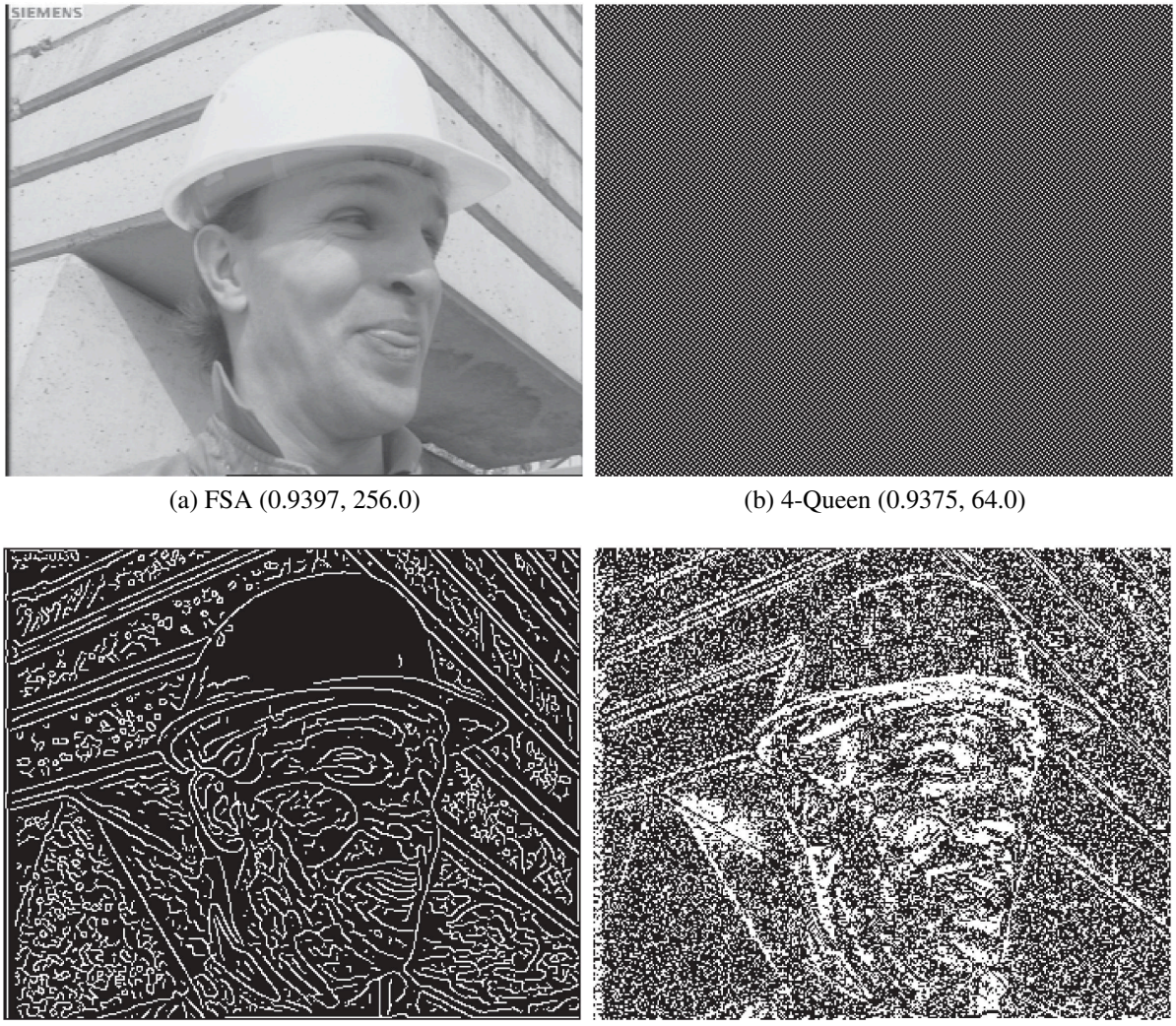
Step 2: Evaluate the $LBP^{D}_{(P,R)}$ codes from $LBP_{(P,R)}$ using look-up table.

Step 3: Obtain the block matching order for each pixel in decreasing order of their difference LBP values.

5.2 Edge Detection Based Motion Estimation (ELBP)

Fast matching approaches essentially try to reduce SAD computations for each block by selecting only a subset of pixels in the block. Conventional works [39] used the assumption that an edge pixel derives the greatest distortion than a background pixel. Therefore, edge pixels play an important role in generating the set of the most representative pixels for a test block. Several filters have been used for edge detection such as FFSSG [39], Roberts, Prewitt, Sobel [21, 51]. These belong to the first-order derivatives and the gradient of an image is obtained through the mask of each method. However, these edge-based techniques are computationally expensive as they need to calculate the magnitude of the gradient for each pixel within a block. The main objective is to develop a technique that reduces the computational complexity significantly while maintaining an acceptable performance. Therefore, instead of calculating the edge gradient for all the pixels, it can be done only for a fraction of them without a significant loss in performance.

N-Queen fixed decimation patterns out-perform other existing patterns in terms of coding efficiency and performance. Each block of size $N \times N$, is decimated by N times to obtain a good



(a) FSA (0.9397, 256.0)

(b) 4-Queen (0.9375, 64.0)

(c) Canny (0.9216, 45.4)

(d) ELBP (0.9331, 26.4)

Figure 5.2: Pixel selections using different approaches (SSIM, NMP) on frame #8 of Foreman sequence (selected pixels are colored white)

representation of the block. Therefore, for a MB of size 16×16, 4-Queen gives us 64 pixels to evaluate. The idea is to reduce the number of pixels further as all of them don't contribute equally to the matching distortion function. Thereby, selecting only the edge pixels using the

LBP operator, their number can be decreased further by making use of the relationship between a pixel and its neighbors to select the most representative pixels. The proposed fast matching pixels for every current block of size 16×16 are identified using the following process:

Step 1: Initially, sixty-four pixels are selected using the 4-Queen fixed decimation pattern.

Step 2: These pixels are evaluated on the basis of their 8-bit LBP code that is generated using (3.3).

Step 3: Only those subset of pixels are selected which belong to the edge.

Fig. 5.2 shows the pixels that are selected using different approaches on frame #8 of Foreman sequence. ELBP is determined only once for the current block, and then this subset pixels can be used for all the candidate blocks within the search range as motion estimation tries to find the best match by comparing with the current reference block. ELBP is a bitwise approach to evaluate the pixels whether they belong to the edge or background, and also doesn't require any 8-bit additions, subtractions or absolute operations. Thus, ELBP can be very efficient for hardware implementation as working with one-bit produces a massive reduction in hardware complexity as one-bit comparator is smaller than an eight-bit accumulator; and in turn reduces power consumption.

5.3 Simulation Results and Analysis

In order to compare the performances of the proposed algorithms, we have used 21 different test sequences. Eight sequences in QCIF (176×144) format (Akiyo, Bus, Carphone, Claire, Container, Grandma, Silent and Suzie), eleven sequences in CIF (352×288) format (Coastguard,

Flower, Football, Foreman, Hall, Miss America, Mobile, Mother & Daughter, Salesman, Table Tennis and Tempete) and two sequence in 720p (1280×720) format (City and Crew) were used in our simulation. These test sequences cover a wide range of motion possibilities and have various different formats (QCIF, CIF and 720p). In our experiments, a block size of 16×16 pixels is used and the search area is ±16 pixels for all the image formats. SAD [as defined in (2.1)] is used as the block matching criterion. All algorithms make comparisons every eight pixels to check if the pSAD has been exceeded. Since, structural similarity (SSIM) [54] metric provides more resemblance to the human visual system as it compares the local patterns of pixel intensities which have been normalized for contrast and luminance instead of intensity difference error-pooling, therefore, SSIM has been used as the quality metric to evaluate the reconstructed images. This section aims to evaluate the effectiveness of the two proposed algorithms.

5.3.1 Performance Evaluation of PLBP

We compare the proposed PLBP with SpiralPDE [1], which is the reference PDE algorithm with spiral matching, FFSSD [39] which represents the full searching technique using matching order based on sorting by distortion, FFSSG [39] which uses matching order based on sorting by gradient, FFSSM [14] which uses matching order based on sorting by mean and HBMO [45] which uses matching order based on sorting by histogram values. These algorithms evaluate all the search positions to find the best match and therefore, their reconstruction quality in terms of SSIM is the same as FSA being lossless. Only by changing the matching order of pixels, they try to reduce the computational complexity. It should be noted that there are additional computational costs in all algorithms to generate the pixel matching order as reported in Table 5.1 except for SpiralPDE. An additional sorting complexity has been referred to as 'c'. Counting sort [16] was used to sort the fixed 256 values to generate the pixel matching order as it is most

efficient in this case. FFSSD does not have any overhead except for the sorting complexity as pixel distortion is calculated in every PDE approach. FFSSG gradient computations is more complex as it calculates the gradient of each pixel in a block. The proposed PLBP requires 2048 comparison operation. The total number of operations for each approach is reported just for complexity comparison purpose. The actual complexity and execution times may vary for each technique depending on the hardware architecture. It is also worth noting that PLBP requires only one type of operator (comparison) which is faster than other arithmetic operators as well as can be used in highly pipelined dedicated hardware.

Table 5.1: Computational complexity for calculating the representative pixels per 16×16 MB for lossless techniques

Operation	**FFSSD**	**FFSSG**	**FFSSM**	**HBMO**	**PLBP**
Addition	–	1792	255	256	–
Subtraction	–	2048	256	–	–
Shift	–	256	1	–	–
Comparison	–	–	–	–	2048
Absolute	–	2048	256	–	–
Sort(c)	once	once	once	once	once
Total	c	6144 + c	768 + c	256 + c	2048 + c

Table 5.2 shows the average number of matched pixels (NMP) per block used to compute the partial distortion. FFSSM yields the largest computational reduction (28.75%) on an average with respect to SpiralPDE followed by the proposed PLBP (28.09%), HBMO (23.88%) and FFSSG (22.30%). The significant improvement of PLBP versus HBMO, FFSSG and FFSSD can be explained by the fact that most contributive pixels can be found easily when the diversity in a local area is considered, that is, a pixel should be selected early if its neighbors are different from each other which has been used in developing the PLBP approach. Performance of FFSSD is worst among these as it does not consider the characteristics of the block and just uses the sorted pixel distortion values.

Table 5.2: Average number of matched pixels (NMP) per 16×16 block using lossless techniques

Format	**Sequence**	**SpiralPDE**	**FFSSD**	**FFSSG**	**FFSSM**	**HBMO**	**PLBP**
QCIF	Akiyo	13.58	10.17	8.64	9.17	9.08	9.45
	Bus	56.53	46.21	43.98	39.08	45.31	39.35
	Carphone	29.43	23.09	21.78	21.65	22.65	21.95
	Claire	16.59	11.94	11.21	12.16	11.38	12.21
	Container	13.61	12.43	10.91	10.52	10.86	10.74
	Grandma	15.26	12.25	11.31	10.68	11.26	10.88
	Silent	19.65	17.93	17.20	16.08	16.91	16.21
	Suzie	29.91	23.28	22.08	20.04	21.14	20.34
	Average	24.32	19.66	18.39	17.42	18.57	17.64
CIF	Coastguard	56.34	50.52	45.11	37.92	42.75	38.22
	Flower	46.30	30.92	27.95	24.38	28.19	25.12
	Football	80.13	69.14	64.64	57.66	62.92	57.81
	Foreman	59.05	48.37	45.28	43.09	43.72	43.32
	Hall	25.23	20.88	19.08	18.50	18.51	18.80
	Miss America	115.40	107.27	100.68	97.62	98.54	97.90
	Mobile	52.28	40.39	35.70	30.76	37.11	31.31
	Mother	65.39	55.39	50.98	49.22	48.63	49.56
	Salesman	43.98	40.64	32.77	29.52	32.83	29.94
	Table Tennis	64.33	54.98	47.59	43.99	46.66	44.19
	Tempete	50.46	43.02	40.04	35.93	41.11	36.30
	Average	59.90	51.05	46.35	42.60	45.54	42.95
720p	City	49.25	38.98	37.06	31.22	35.53	31.67
	Crew	67.00	55.28	54.66	51.59	52.96	51.86
	Average	58.13	47.13	45.86	41.40	44.25	41.77

5.3.2 Performance Evaluation of ELBP

The proposed ELBP approach has been compared with FSA, 4-Queen [52], Roberts [21], Prewitt [21], Sobel [21], Canny [5], B4Q4 [49] and BVG11 [49]. Canny [5] is an edge detection algorithm that uses a multi-stage technique to detect a wide range of edges in images. But, it is computationally very expensive as it involves various stages of detection, localization and minimizing multiple responses to a single edge. It has been used just to compare the performance of ELBP in terms of reconstruction image quality and number of matching pixels. Its computations are not considered because of the significant overheads.

Table 5.3 shows the comparison of the number of operations required for each edge detection algorithm per 16×16 MB. It must be noted that these are extra computations required in order to evaluate the most contributive edge pixels. 4-Queen, B4Q4 and BVG11 do not require any added computations as they are fixed-pixel decimation approaches. It can be seen that ELBP does not have much overheads in terms of total number of operations. It reduces the computational complexity by 60% and 91.67% over Roberts and FFSSG, respectively. Also since, ELBP edges are derived using only a comparison operator which is faster than addition operator, it performs edge detection faster than other methods.

Table 5.3: Computational complexity for calculating the representative pixels per 16×16 MB for lossy techniques

Operation	**Roberts**	**Prewitt**	**Sobel**	**FFSSG**	**ELBP**
Addition	256	2304	2304	1792	–
Subtraction	512	512	512	2048	–
Shift	–	–	1024	256	–
Comparison	–	–	–	–	512
Absolute	512	512	512	2048	–
Total	1280	3328	4352	6144	512

Table 5.4 shows the average number of matched pixels per block using various algorithms. It can be seen ELBP consistently gives the minimum NMP per block for all the test sequences. Also its last column under ELBP shows the computational reduction (%) in terms of the number of matched pixels with respect to 4-Queen. In average, the ELBP reduces the computational complexity of 4-Queen by 62.67% with other techniques always performing worse.

Table 5.5 and Table 5.6 show the performance evaluation of the ELBP with respect to various algorithms using the SSIM and PSNR as the quality metric, respectively. On an average for all test sequences, Prewitt and Sobel have the worst performance when the number of search pixels is close to sixty-four per block, especially for Football and Salesman sequences. This is because the fraction of pixels selected for matching are not sufficient for good reconstruction quality. At about sixty-four pixels per block, 4-Queen performs the best among Roberts, Prewitt and Sobel. ELBP outperforms state-of-the-art Canny approach in terms of reconstruction quality with even lesser number of matching pixels ($\sim 54\%$ less pixels). ELBP also performs close to B4Q4 and BVG11 algorithms given that the reduction in NMP is 40% and 34% respectively.

Table 5.4: Average number of matched pixels (NMP) per 16×16 block using lossy techniques and computation reduction (%) of ELBP over 4-Queen is shown in the last column under ELBP

Format	Sequence	FSA	4-Queen	Roberts	Prewitt	Sobel	Canny	B4Q4	BVG11	ELBP	
QCIF	Akiyo	256.0	64.0	44.9	40.7	40.9	47.4	40.0	36.0	25.9	59.53
	Bus	256.0	64.0	72.2	77.4	76.2	59.9	40.0	36.0	19.2	69.97
	Carphone	256.0	64.0	49.2	43.5	43.7	40.3	40.0	36.0	29.0	54.76
	Claire	256.0	64.0	22.7	19.1	19.1	29.7	40.0	36.0	27.4	57.22
	Container	256.0	64.0	44.7	46.9	47.4	47.8	40.0	36.0	23.2	63.69
	Grandma	256.0	64.0	54.1	49.8	49.7	46.6	40.0	36.0	25.1	60.79
	Silent	256.0	64.0	68.9	63.6	63.9	49.8	40.0	36.0	23.7	62.95
	Suzie	256.0	64.0	48.0	42.0	42.4	42.1	40.0	36.0	28.3	55.73
	Average	256.0	64.0	50.6	47.9	47.9	45.4	40.0	36.0	25.2	60.58
CIF	Coastguard	256.0	64.0	83.8	88.0	88.2	65.3	40.0	36.0	22.9	64.28
	Flower	256.0	64.0	50.5	57.2	56.2	43.7	40.0	36.0	12.9	79.90
	Football	256.0	64.0	73.0	85.9	85.2	63.7	40.0	36.0	23.8	62.81
	Foreman	256.0	64.0	55.0	51.6	51.7	48.6	40.0	36.0	25.7	59.84
	Hall	256.0	64.0	33.0	32.7	32.7	39.3	40.0	36.0	25.9	59.52
	Miss America	256.0	64.0	39.9	21.4	21.7	52.7	40.0	36.0	20.7	67.65
	Mobile	256.0	64.0	65.0	69.9	68.9	56.7	40.0	36.0	26.8	58.13
	Mother	256.0	64.0	39.9	36.1	36.8	49.0	40.0	36.0	24.8	61.32
	Salesman	256.0	64.0	67.2	63.2	62.8	53.4	40.0	36.0	23.3	63.54
	Table Tennis	256.0	64.0	65.7	80.4	80.1	69.8	40.0	36.0	21.5	66.41
	Tempete	256.0	64.0	62.6	62.8	62.0	51.7	40.0	36.0	22.4	65.01
	Average	256.0	64.0	57.8	59.0	58.8	54.0	40.0	36.0	22.8	64.40
720p	City	256.00	64.00	69.80	71.53	70.93	60.56	40.00	36.00	19.76	69.12
	Crew	256.00	64.00	33.65	32.06	32.23	45.23	40.00	36.00	27.56	56.93
	Average	256.00	64.00	51.73	51.80	51.58	52.90	40.00	36.00	23.66	63.03

Table 5.5: Performance Comparison in SSIM for lossy techniques. Actual SSIM values are reported for FSA and 4-Queen while SSIM difference (obtained by subtracting the SSIM of an approach from the SSIM of 4-Queen) is reported for other techniques

Format	**Sequence**	**FSA**	**4-Queen**	**Roberts**	**Prewitt**	**Sobel**	**Canny**	**B4Q4**	**BVG11**	**ELBP**
QCIF	Akiyo	0.9943	0.9943	0.0008	0.0009	0.0007	0.0001	0.0001	0.0001	0.0002
	Bus	0.8814	0.8786	0.0220	0.0285	0.0294	0.0343	0.0043	0.0067	0.0203
	Carphone	0.9480	0.9469	0.0148	0.0139	0.0141	0.0217	0.0020	0.0025	0.0054
	Claire	0.9940	0.9939	0.0020	0.0024	0.0027	0.0023	0.0001	0.0002	0.0005
	Container	0.9933	0.9932	0.0028	0.0037	0.0037	0.0005	0.0001	0.0002	0.0006
	Grandma	0.9915	0.9914	0.0010	0.0012	0.0013	0.0012	0.0001	0.0001	0.0005
	Silent	0.9647	0.9641	0.0049	0.0047	0.0047	0.0078	0.0011	0.0013	0.0037
	Suzie	0.9648	0.9642	0.0076	0.0074	0.0072	0.0065	0.0009	0.0012	0.0036
	Average	0.9665	0.9658	0.0070	0.0078	0.0080	0.0093	0.0011	0.0015	0.0043
CIF	Coastguard	0.9145	0.9133	0.0040	0.0053	0.0051	0.0080	0.0024	0.0036	0.0068
	Flower	0.9338	0.9314	0.0128	0.0156	0.0162	0.0136	0.0036	0.0044	0.0194
	Football	0.7779	0.7705	0.0166	0.0217	0.0217	0.0338	0.0105	0.0138	0.0198
	Foreman	0.9231	0.9210	0.0155	0.0237	0.0237	0.0174	0.0033	0.0042	0.0088
	Hall	0.9880	0.9878	0.0086	0.0095	0.0094	0.0061	0.0004	0.0006	0.0031
	Miss America	0.9242	0.9188	0.0082	0.0254	0.0277	0.0078	0.0032	0.0044	0.0090
	Mobile	0.9052	0.9026	0.0163	0.0263	0.0276	0.0203	0.0041	0.0063	0.0162
	Mother	0.9698	0.9689	0.0081	0.0071	0.0071	0.0078	0.0009	0.0011	0.0026
	Salesman	0.9413	0.9405	0.0073	0.0129	0.0125	0.0059	0.0012	0.0012	0.0059
	Table Tennis	0.8964	0.8865	0.0007	0.0059	0.0067	0.0021	0.0001	0.0010	0.0038
	Tempete	0.9078	0.9057	0.0195	0.0232	0.0239	0.0224	0.0032	0.0039	0.0106
	Average	0.9165	0.9134	0.0107	0.0161	0.0165	0.0132	0.0030	0.0041	0.0096
720p	City	0.9662	0.9656	0.0109	0.0215	0.0214	0.0125	0.0010	0.0018	0.0063
	Crew	0.9390	0.9380	0.0369	0.0455	0.0447	0.0330	0.0023	0.0028	0.0085
	Average	0.9526	0.9518	0.0239	0.0335	0.0330	0.0228	0.0016	0.0023	0.0074

Table 5.6: Performance Comparison in PSNR (dB) for lossy techniques

Format	Sequence	FSA	4-Queen	Roberts	Prewitt	Sobel	Canny	B4Q4	BVG11	ELBP
QCIF	Akiyo	43.60	43.59	43.29	43.26	43.37	43.57	43.58	43.54	43.49
	Bus	24.11	24.02	22.79	22.72	22.70	22.04	23.84	23.73	22.98
	Carphone	31.47	31.38	29.80	29.92	29.95	28.53	31.21	31.18	30.70
	Claire	41.85	41.82	40.55	40.57	40.38	40.33	41.77	41.75	41.58
	Container	41.07	41.06	40.14	39.93	39.93	40.34	41.03	40.98	40.64
	Grandma	42.95	42.93	42.58	42.46	42.29	42.16	42.90	42.87	42.75
	Silent	34.31	34.25	33.71	33.87	33.90	33.30	34.13	34.13	33.82
	Suzie	35.42	35.36	34.40	34.85	34.83	34.39	35.28	35.23	34.93
	Average	36.85	36.80	35.91	35.95	35.92	35.58	36.72	36.68	36.36
CIF	Coastguard	30.48	30.39	30.03	30.08	30.08	29.54	30.28	30.23	29.97
	Flower	25.23	25.11	24.42	24.49	24.40	24.16	24.95	24.92	24.06
	Football	23.68	23.52	22.75	22.97	22.98	22.22	23.25	23.23	22.73
	Foreman	32.12	32.01	30.31	30.68	30.70	30.08	31.85	31.81	31.38
	Hall	36.22	36.17	34.02	33.87	33.90	33.79	36.03	35.99	34.98
	Miss America	37.77	37.53	36.79	34.40	34.12	35.27	37.32	37.25	34.98
	Mobile	24.62	24.50	23.70	23.48	23.43	23.31	24.32	24.23	23.72
	Mother	40.47	40.37	37.17	38.48	38.39	35.75	40.25	40.22	39.79
	Salesman	35.58	35.50	34.69	34.81	34.77	33.96	35.29	34.31	34.08
	Table Tennis	29.22	29.06	28.50	28.23	28.19	28.22	28.85	28.77	28.14
	Tempete	26.22	26.14	25.24	25.13	25.12	25.00	26.00	25.98	25.62
	Average	31.06	30.94	29.78	29.69	29.64	29.21	30.76	30.63	29.95
720p	City	31.88	31.80	30.91	30.35	30.35	30.56	31.70	31.61	30.95
	Crew	33.78	33.70	30.75	30.73	30.77	30.35	33.54	33.50	33.06
	Average	32.83	32.75	30.83	30.54	30.56	30.46	32.62	32.55	32.01

Table 5.7: Performance evaluation of ELBP with respect to 4-Queen algorithm in terms of SSIM difference and NMP computational reduction

Sequence	**$\triangle$SSIM**	**$\%$ Red.**
Foreman	-0.0088	59.84
Football	-0.0198	62.81
Tennis	-0.0038	66.41
Coastguard	-0.0068	64.22
Akiyo	-0.0002	59.53
Mother	-0.0026	58.12
Salesman	-0.0059	63.59
Average	-0.0069	62.03

Table 5.7 shows the performance evaluation of the ELBP with respect to 4-Queen in terms of SSIM difference and computational reduction in terms of the number of matched pixels. It should be noted that SSIM difference is obtained by subtracting the SSIM of the 4-Queen from the SSIM of ELBP. In average, the ELBP reduces the computational complexity of 4-Queen by 62% while maintaining comparable performance.

The ELBP can reduce a large amount of computations because of its good edge detection characteristics. It selects only those representative pixels which are likely to contribute highly to the distortion matching function. The most contributable pixels are found mostly in the edges which can be easily identified using this approach as compared to other conventional methods. Therefore, ELBP provides an excellent performance to cost ratio as compared to other methods for all test sequences.

Chapter 6

Conclusion

In Chapter 4, a simple and fast neighborhood-matching algorithm based on local binary patterns was introduced. By exploiting the neighborhood information around a pixel in a local region, it effectively captures the salient micro-features. In addition, it is also robust in terms of gray-scale variations. FS-LBP makes use of fast binary matching criterion for reducing the computational complexity of matching error function. It results in better performance in terms of SSIM than even full search algorithm. A low complexity approach (RC-LBP) was also proposed which reduces the computational complexity hugely. It makes use of average motion vector to reduce the problem of blocking artifacts when the size of macro-block is large. When compared to 1BT, 1BT-MF and 2BT-SD, RC-LBP significantly increases the computational gain with a marginal loss in quality criterion metrics but, it gives better quality visually. Also, RC-LBP's simplicity and regularity are very desirable and attractive for hardware implementations. Therefore, RC-LBP is a suitable candidate for implementation on mobile terminals owing to its low computational complexity and acceptable performance.

In Chapter 5, we investigated a couple of edge detection based partial distortion elimination

approaches. The first algorithm is a lossless fast motion estimation PDE algorithm based on local binary patterns (PLBP). An analysis on the characteristics of pixel-to-pixel distortion has also been presented. PLBP reduces the computational complexity over conventional methods by generating a good matching order based on the non-uniformity around a pixel. The second approach is a lossy motion estimation algorithm based on edge-detection using local binary patterns (ELBP). This approach selects only the most contributable pixels to the distortion function for matching by identifying the edge pixels using LBP. Experimental results demonstrate substantial reduction in computational complexity of ELBP with only a marginal loss in prediction quality.

6.1 Future Work

The neighborhood-matching algorithm discussed in Chapter 4 attempts to take advantage of both reduction in search positions and bitwidth reduction while edge-detection approach mentioned in Chapter 5 tries to unify reduction in matching pixels and bitwidth reduction. In future, we plan to integrate these algorithms together to investigate if there is further improvement in computational complexity while maintaining performance on par or better than other approaches.

References

[1] "ITU-T Recommendation H.263 Software Implementations", Digital Video Coding Group, Telenor R&D, 1995.

[2] A. Akin, Y. Dogan, and I. Hamzaoglu. High performance hardware architectures for one bit transform based motion estimation. *Consumer Electronics, IEEE Transactions on*, 55(2):941–949, 2009.

[3] Aroh Barjatya. Block matching algorithms for motion estimation. *Final Project Paper for Spring 2004 Digital Image Processing Course at the Utah State University*, 2004.

[4] M. Bierling. Displacement estimation by hierarchical block matching. *Proc. of SPIE Visuak Commun. Image Processing(VCIP'88)*, pages 449–452, 1988.

[5] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679 –698, nov. 1986.

[6] A. Celebi, O. Urhan, I. Hamzaoglu, and S. Erturk. Efficient hardware implementations of low bit depth motion estimation algorithms. *Signal Processing Letters, IEEE*, 16(6):513–516, 2009.

[7] Y.-L. Chan and W.-C. Siu. Edge oriented block motion estimation for video coding. *Vision, Image and Signal Processing, IEE Proceedings* -, 144(3):136 –144, jun 1997.

[8] Yui-Lam Chan, Wai-Lam Hui, and Wan-Chi Siu. A block motion vector estimation using pattern based pixel decimation. In *Circuits and Systems, 1997. ISCAS '97., Proceedings of 1997 IEEE International Symposium on*, volume 2, pages 1153 –1156 vol.2, jun 1997.

[9] Yui-Lam Chan and Wan-Chi Siu. A new block motion vector estimation using adaptive pixel decimation. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 4, pages 2257 –2260 vol.4, may 1995.

[10] Yui-Lam Chan and Wan-Chi Siu. New adaptive pixel decimation for block motion vector estimation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 6(1):113 –118, feb 1996.

[11] Yui-Lam Chan and Wan-Chi Siu. An efficient search strategy for block motion estimation using image features. *Image Processing, IEEE Transactions on*, 10(8):1223 –1238, aug 2001.

[12] Mei-Juan Chen, Liang-Gee Chen, Tzi-Dar Chiueh, and Yung-Pin Lee. A new block-matching criterion for motion estimation and its implementation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 5(3):231 –236, jun 1995.

[13] Man-Yau Chiu and Wan-Chi Siu. Computationally-scalable motion estimation algorithm for h.264/avc video coding. *Consumer Electronics, IEEE Transactions on*, 56(2):895–903, 2010.

[14] Changryoul Choi and Jechang Jeong. New sorting-based partial distortion elimination algorithm for fast optimal motion estimation. *Consumer Electronics, IEEE Transactions on*, 55(4):2335 –2340, november 2009.

[15] Changryoul Choi and Jechang Jeong. Enhanced two-bit transform based motion estimation via extension of matching criterion. *Consumer Electronics, IEEE Transactions on*, 56(3):1883 –1889, aug. 2010.

[16] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[17] S. Dikbas, T. Arici, and Y. Altunbasak. Fast motion estimation with interpolation-free sub-sample accuracy. *Circuits and Systems for Video Technology, IEEE Transactions on*, 20(7):1047–1051, 2010.

[18] A. Erturk and S. Erturk. Two-bit transform for binary block motion estimation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 15(7):938 – 946, july 2005.

[19] S. Erturk. Multiplication-free one-bit transform for low-complexity block-based motion estimation. *Signal Processing Letters, IEEE*, 14(2):109 –112, feb. 2007.

[20] H. Gharavi and M. Mills. Blockmatching motion estimation algorithms-new results. *Circuits and Systems, IEEE Transactions on*, 37(5):649 –651, may 1990.

[21] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3nd edition, 2007.

[22] Y. Ismail, J.B. McNeely, M. Shaaban, H. Mahmoud, and M.A. Bayoumi. Fast motion estimation system using dynamic models for h.264/avc video coding. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(1):28–42, 2012.

[23] Jong-Nam Kim, Sung-Cheal Byun, and Byung-Ha Ahn. Fast full search motion estimation algorithm using various matching scans in video coding. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 31(4):540 –548, nov 2001.

[24] Jong-Nam Kim, Sung-Cheal Byun, Yong-Hoon Kim, and Byung-Ha Ahn. Fast full search motion estimation algorithm using early detection of impossible candidate vectors. *Signal Processing, IEEE Transactions on*, 50(9):2355 – 2365, sep 2002.

[25] Jong-Nam Kim and Tae-Sun Choi. Adaptive matching scan algorithm based on gradient magnitude for fast full search in motion estimation. *Consumer Electronics, IEEE Transactions on*, 45(3):762 –772, aug 1999.

[26] Jong-Nam Kim and Tae-Sun Choi. A fast full-search motion-estimation algorithm using representative pixels and adaptive matching scan. *Circuits and Systems for Video Technology, IEEE Transactions on*, 10(7):1040 –1048, oct 2000.

[27] Nam-Joon Kim, S. Erturk, and Hyuk-Jae Lee. Two-bit transform based block motion estimation using second derivatives. *Consumer Electronics, IEEE Transactions on*, 55(2):902 –910, may 2009.

[28] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro. Motion compensated interframe coding for video conferencing. *Nat. Telecommun. Conf.*, pages G5.3.1 –G5.3.5, 1981.

[29] Peter M. Kuhn. *Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*. Kluwer Academic Publishers, Norwell, MA, USA, 1st edition, 1999.

[30] C.J. Kuo, C.H. Yeh, and S.F. Odeh. Polynomial search algorithms for motion estimation. In *Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on*, volume 4, pages 215 –218 vol.4, jul 1999.

[31] Hyuk Lee and Jechang Jeong. Early termination scheme for binary block motion estimation. *Consumer Electronics, IEEE Transactions on*, 53(4):1682 –1686, nov. 2007.

[32] Xiaobing Lee and Ya-Qin Zhan. A fast hierarchical motion-compensation scheme for video coding using block feature matching. *Circuits and Systems for Video Technology, IEEE Transactions on*, 6(6):627 –635, dec 1996.

[33] Reoxiang Li, Bing Zeng, and M.L. Liou. A new three-step search algorithm for block motion estimation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 4(4):438 –442, aug 1994.

[34] Jinho Lim and Jechang Jeong. An efficient search method for binary-based motion estimation. In *Consumer Electronics (ISCE), 2011 IEEE 15th International Symposium on*, pages 132 –137, june 2011.

[35] Weiyao Lin, K. Panusopone, D.M. Baylon, Ming-Ting Sun, Zhenzhong Chen, and Hongxiang Li. A fast sub-pixel motion estimation algorithm for h.264/avc video coding. *Circuits and Systems for Video Technology, IEEE Transactions on*, 21(2):237–242, 2011.

[36] B. Liu and A. Zaccarin. New fast algorithms for the estimation of block motion vectors. *Circuits and Systems for Video Technology, IEEE Transactions on*, 3(2):148 –157, apr 1993.

[37] Jeng-Hung Luo, Chung-Neng Wang, and Tihao Chiang. A novel all-binary motion estimation (abme) with optimized hardware architectures. *Circuits and Systems for Video Technology, IEEE Transactions on*, 12(8):700 – 712, aug 2002.

[38] T. Mäenpää. *The Local Binary Pattern Approach to Texture Analysis: Extensions and Applications*. Acta Universitatis Ouluensis: Technica. Oulun yliopisto, 2003.

[39] B. Montrucchio and D. Quaglia. New sorting-based lossless motion estimation algorithms and a partial distortion elimination performance analysis. *Circuits and Systems for Video Technology, IEEE Transactions on*, 15(2):210 – 220, feb. 2005.

[40] Kwon Moon Nam, Joon-Seek Kim, Rae-Hong Park, and Young Serk Shim. A fast hierarchical motion vector estimation algorithm using mean pyramid. *Circuits and Systems for Video Technology, IEEE Transactions on*, 5(4):344 –351, aug 1995.

[41] B. Natarajan, V. Bhaskaran, and K. Konstantinides. Low-complexity block-based motion estimation via one-bit transforms. *Circuits and Systems for Video Technology, IEEE Transactions on*, 7(4):702 –706, aug 1997.

[42] A.C.K. Ng and B. Zeng. A new fast motion estimation algorithm based on search window sub-sampling and object boundary pixel block matching. In *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, pages 605 –608 vol.3, oct 1998.

[43] Yao Nie and Kai-Kuang Ma. Adaptive rood pattern search for fast block-matching motion estimation. *Image Processing, IEEE Transactions on*, 11(12):1442 – 1449, dec 2002.

[44] T. Ojala, M. Pietikainen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971 –987, jul 2002.

[45] S.-J. Park, S.-M. Hong, H. Lee, S. Jin, and J. Jeong. Histogram ordering model-based fast motion estimation. *Image Processing, IET*, 6(3):238 –250, april 2012.

[46] A. Paul, J. Wu, J. F Yang, and J. Jeong. Gradient-based edge detection for motion estimation in h.264/avc. *Image Processing, IET*, 5(4):323–327, 2011.

[47] Lai-Man Po and Wing-Chung Ma. A novel four-step search algorithm for fast block motion estimation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 6(3):313 – 317, jun 1996.

[48] John B. Reppas, Sourabh Niyogi, Anders M. Dale, Martin I. Sereno, and Roger B. Tootell. Representation of motion boundaries in retinotopic human visual cortical areas. *Nature*, 388(6638):175–179, July 1997.

[49] Avishek Saha, Jayanta Mukherjee, and Shamik Sural. New pixel-decimation patterns for block matching in motion estimation. *Image Commun.*, 23(10):725–738, November 2008.

[50] Liquan Shen, Zhi Liu, Tao Yan, Zhaoyang Zhang, and Ping An. View-adaptive motion estimation and disparity estimation for low complexity multiview video coding. *Circuits and Systems for Video Technology, IEEE Transactions on*, 20(6):925–930, 2010.

[51] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[52] Chung-Neng Wang, Shin-Wei Yang, Chi-Min Liu, and Tihao Chiang. A hierarchical decimation lattice based on n-queen with an application for motion estimation. *Signal Processing Letters, IEEE*, 10(8):228 – 231, aug. 2003.

[53] Yankang Wang, Yanqun Wang, and H. Kuroda. A globally adaptive pixel-decimation algorithm for block-motion estimation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 10(6):1006 –1011, sep 2000.

[54] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600 –612, april 2004.

[55] Zhou Wang and Qiang Li. Information content weighting for perceptual image quality assessment. *Image Processing, IEEE Transactions on*, 20(5):1185–1198, 2011.

[56] P.H.W. Wong and O.C. Au. Modified one-bit transform for motion estimation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 9(7):1020 –1024, oct 1999.

[57] Wei Zhou, Zhemin Duan, and Hongqi Hu. Fast motion estimation algorithm for h.264/avc based on centered prediction. *Systems Engineering and Electronics, Journal of*, 21(6):1103–1110, 2010.

[58] Ce Zhu, Xiao Lin, and L-P Chau. Hexagon-based search pattern for fast block motion estimation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 12(5):349–355, 2002.

[59] Shan Zhu and Kai-Kuang Ma. A new diamond search algorithm for fast block-matching motion estimation. *Image Processing, IEEE Transactions on*, 9(2):287 –290, feb 2000.