Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2008, Article ID 180582, 10 pages
doi:10.1155/2008/180582

Research Article

Probabilistic Global Motion Estimation Based on Laplacian Two-Bit Plane Matching for Fast Digital Image Stabilization

Nam-Joon Kim,[1] Hyuk-Jae Lee,[1] and Jae-Beom Lee[2]

[1] *Inter-University Semiconductor Research Center (ISRC), Department of Electrical Engineering and Computer Science, Seoul National University, Seoul 151-744, South Korea*
[2] *Sarnoff Corporation, P.O. Box 5300, 201 Washington Road, Princeton, NJ 08543, USA*

Correspondence should be addressed to Hyuk-Jae Lee, hjlee_paper@capp.snu.ac.kr

Received 25 May 2007; Revised 11 November 2007; Accepted 21 December 2007

Recommended by D. O'Shaughnessy

Digital image stabilization (DIS) is a technique to prevent images captured by a handheld camera from temporal fluctuation. This paper proposes a new DIS algorithm that reduces the computation time while preserving the accuracy of the algorithm. To reduce the computation time, an image is transformed by a Laplacian operation and then converted into two one-bit spaces, called L^+ and L^- spaces. The computation time is reduced because only two-bits-per-pixel are used while the accuracy is maintained because the Laplacian operation preserves the edge information which can be efficiently used for the estimation of camera motion. Either two or four subimages in the corners of an image frame are selected according to the type of the image and five local motion vectors with their probabilities to be a global motion vector are derived for each subimage. The global motion vector is derived from these local motion vectors based on their probabilities. Experimental results show that the proposed algorithm achieves a similar or better accuracy than a conventional DIS algorithm using a local motion estimation based on a full-search scheme and MSE criterion while the complexity of the proposed algorithm is much less than the conventional algorithm.

1. INTRODUCTION

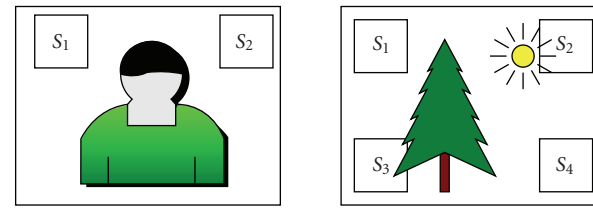
Digital image stabilization (DIS) is a technique to compensate an irregular camera motion in a captured video sequence, and obtain a stable video sequence with smooth camera motion [1–7]. By reducing abrupt motion between successive image frames, DIS improves the compression efficiency when a video sequence is encoded based on a compression standard such as MPEG or H.264 [8, 9]. It can also eliminate remnant images on LCD screen due to high-frequency jitters of images and the slow reaction of the LCD screen [1]. As DIS utilizes only digital image processing techniques, it can be easily integrated with other digital logics in a single chip. As a result, the implementation cost of DIS is very low when compared with conventional optical image stabilization (OIS) that uses mechanical devices like a gyro sensor or a fluid prism.

In order to compensate an undesirable camera motion, a DIS system, in general, derives the global motion vector (GMV) which represents the motion between the current image frame and the previous image frame. One of the widely-used algorithms to obtain the GMV is the full-search frame matching (FS-FM) motion estimation in which an entire frame is used to compare the current and previous frames and the best-matched displacement between the two frames is chosen as the GMV [3, 4, 10]. Another popular algorithm is full-search block-based matching (FS-BM) in which the current frame is divided into many blocks and the motion is estimated for each block. This estimated motion for each block is represented by a local motion vector (LMV). By combining the LMVs derived for all the blocks in a frame, the GMV of the frame is derived [2, 7, 11].

Mean absolute difference (MAD) or mean square error (MSE) is generally used as the criterion of how well the

current frame is matched with the previous frame [11]. The amount of computation required by FS-FM or FS-BM using MAD or MSE is very large as reported in [1, 3, 4]. Hence, various algorithms have been proposed to reduce the computation for motion estimation [1, 3–6, 10–14]. One approach to reduce the computation is to reduce the size of blocks for which LMVs are derived [5, 10]. In this approach, small blocks in the four corners of an image frame are, in general, chosen for the derivation of LMVs and the GMV is derived based on these four LMVs. Note that there is a high possibility that the corners of an image is a background area of which the motion should be compensated by DIS while the movement of foreground objects should be preserved even with DIS. Another approach for computation reduction is to reduce the number of pixels in a block for motion estimation. In [6], the edge pattern of an image is derived and only edge regions are compared for the best match between frames. This method reduces the computation at the expense of the accuracy of motion estimation. Fast motion estimation methods based on bit-plane or gray-coded bit-plane matching have been proposed in [3, 4], respectively. These approaches reduce the number of bits to represent one pixel, resulting in the reduction of the computation while maintaining the motion estimation accuracy. Another method that obtains the motion vector using a binary operation is one-bit transform (1BT)-based motion estimation in which image frames are transformed into a single bit-plane after comparing the original image frame against its multiband-pass filtered version [10]. This method also provides a low computational complexity with reasonably accurate motion estimation results. Subimage phase-correlation-based global motion estimation is proposed in [1, 15]. This algorithm generates relatively accurate motion estimation results, but requires large computational complexity due to the computation for 2D-Fourier transforms [10]. Recently, a digital image stabilizer integrated with a video codec has been proposed in [16]. One of the three schemes proposed in this research is a technique to reduce the computational complexity of the digital stabilizer by using the information obtained by the motion estimation in the video codec.

This paper proposes a novel DIS algorithm that outperforms previously proposed algorithms in terms of motion estimation accuracy and computational complexity. For LMV derivation, either two or four subimages are chosen in the corner of an image frame. To obtain LMVs, the second derivatives by Laplacian operation of subimages are computed and the computation results are transformed into two-bit-per-pixel representation. This two-bit representation called Laplacian two-bit transform (L2BT) contributes to the computation reduction due to the decrease of the number of bits per pixel, while the motion estimation accuracy is still maintained because the edge information is preserved by Laplacian operations. An LMV for each subimage is derived with binary block matching with the corresponding subimage in the previous frame. This derivation also considers the distance of the derived LMV from the previous global motion vector (PGMV). For each subimage, five candidate LMVs with the largest L2BT block matching and the smallest distance from PGMV are derived. From LMVs obtained for subimages, a novel algorithm to obtain the GMV is proposed. Although this algorithm for GMV derivation is more complex than a conventional method, the increase of the computation time of overall DIS algorithm is marginal because the GMV derivation algorithm has much less complexity than the search process for local motion estimation.



(a) Subimages for single human image (b) Subimages for general image

Figure 1: The subimages for the LMV estimation.

This paper is organized as follows. Section 2 presents the L2BT block-based correlation matching algorithm for the derivation of LMVs. In Section 3, the GMV derivation algorithm is proposed. Experimental results compared with prior DIS algorithms are given in Section 4 and conclusions are drawn in Section 5.

2. LOCAL MOTION ESTIMATION BASED ON LAPLACIAN TWO-BIT PLANE MATCHING

2.1. Selection of subimages for local motion estimation

This paper uses two subimages S_1, S_2 as shown in Figure 1(a) for the derivation of LMVs for images with a single human as the only foreground object while it utilizes four subimages S_1, S_2, S_3, and S_4 as shown in Figure 1(b) for generic scenes. A single human image may be often used for video telephony communication while the amount of computation can be reduced with two subimages instead of four subimages.

2.2. Feature extraction

To derive an LMV, the second derivatives of an original image are derived because it can represent edge information while ignoring flat areas. As a result, a motion can be effectively estimated with edge areas while computation load can be reduced with the ignorance of flat areas. The equation used to obtain the second derivatives for image $f(x, y)$ is as follows:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}. \tag{1}$$

The digital realization of (1) is as follows [17]:

$$\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y). \tag{2}$$

The Laplacian masks as shown in Figure 2 can be used for the derivation of (2). Either of the two masks shown Figure 2 can be used for the second derivative operation and

this paper uses the left smask in Figure 2. The second derivative operation using Laplacian mask needs one shift and four add/sub operations for one pixel. Note that the Laplacian mask operation requires less computation than the first derivative operation by Sobel masks that requirefour shift and eleven add/sub operations for one pixel [17].

If the Laplacian mask operation results in very small values for certain pixels, these pixels may not be in an edge area. Therefore, it may be reasonable to ignore these results for motion estimation. To this end, a threshold value is chosen so that some Laplacian results are converted to 0 if their absolute values are less than the threshold. Based on experimental results with sample video sequences, $1/2^5$ of maximum positive value among all results is chosen as the positive threshold while $1/2^5$ of the minimum negative value is chosen as the negative threshold. This conversion to 0 can also help to reduce the computation load. The laplacian operation results are expressed in two spaces, called L^+ and L^- spaces. When a Laplacian operation result is positive and greater than or equal to the positive threshold, 1 is stored in the L^+ space. On the other hand, when it is negative and less than or equal to the negative threshold, 1 is stored in the L^- space. For all the remaining pixels, 0 is stored in both the L^+ and L^- spaces. Note that the number of pixels in each space is the same as the original image.

As the measure to estimate the correlation between the two subimages in the current frame and the previous frame, the number of nonmatching points (NNMP) [18–20] is used. NNMP is defined as a distance in this paper as follows:

$$D(i,j) = \sum_{(x,y)\in s} \left\{L_t^+(x,y) \oplus L_{t-1}^+(x+i,y+j)\right\} \cup \left\{L_t^-(x,y) \oplus L_{t-1}^-(x+i,y+j)\right\}, \quad \text{for } -r \le i,\ j \le r, \tag{3}$$

where s represents the range of pixels for which the distance is calculated. The detailed explanation about this range is given in Section 2.4. In (3), r represents the search range, and $L_t^+(x,y)$ and $L_{t-1}^+(x,y)$ denote the values of pixel (x,y) in the L^+ space at the current and previous image frames, respectively. $L_t^-(x,y)$ and $L_{t-1}^-(x,y)$ represent the corresponding pixel values in the L^- space. Symbols $\oplus$ and $\cup$ represent Boolean exclusive-Or (XOR) and Boolean-Or operations, respectively.

2.3. LMV selection and probability estimation

Generally, one LMV is derived for one subimage. However, in this paper, five LMV candidates are derived for a subimage. In addition, the probability of each LMV candidate to be the GMV is also assigned. This section explains the method for obtaining five LMVs with the probabilities.

The probability for a specific vector (i,j) to be the GMV is denoted by $P_c(i,j)$. Note that the summation of $P_c(i,j)$ for all pixels (i,j) in the search range must be equal to 1 because $P_c(i,j)$ is a probability distribution function. Thus, the following equation must be satisfied:

$$\sum_{-r\le i,j\le r} P_c(i,j) = 1. \tag{4}$$

0	1	0
1	−4	1
0	1	0

or

0	−1	0
−1	4	−1
0	−1	0

Figure 2: Laplacian masks for second derivative operation.

As $D(i,j)$ of (3) decreases, $P_c(i,j)$ becomes large. Another independent variable determining the probability for a vector (i,j) to be the GMV is the distance from PGMV. This is because a hand-movement is relatively slower than the frame rate of a camera and the motion vector of the current frame is often almost the same as that of the previous frame [4]. If multiple vectors have the same $D(i,j)$ value, the vector near PGMV should have a high probability to be the GMV. Let (g_x, g_y) denote the PGMV and $D_{\text{PGMV}}(i,j)$ denote the distance of a vector from PGMV

$$D_{\text{PGMV}}(i,j) = \sqrt{(i-g_x)^2 + (j-g_y)^2}. \tag{5}$$

Then, $P_c(i,j)$ increases as either $D(i,j)$ decreases or $D_{\text{PGMV}}(i,j)$ decreases. Let $f(D(i,j))$ and $g(D_{\text{PGMV}}(i,j))$ denote monotonically nondecreasing functions of $D(i,j)$ and $D_{\text{PGMV}}(i,j)$, respectively. In order to reduce the computational load, this paper chooses simple models of functions $f(D(i,j))$ and $g(D_{\text{PGMV}}(i,j))$. The proposed models simply use the aspect that the functions $f(D(i,j))$ and $g(D_{\text{PGMV}}(i,j))$ are monotonically nondecreasing

$$f(D(i,j)) = D(i,j) - (\min(D(i,j)) - \alpha), \tag{6}$$

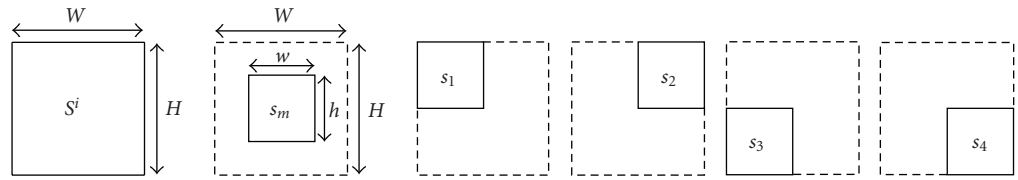
$$g(D_{\text{PGMV}}(i,j)) = \begin{cases} \beta D_{\text{PGMV}}(i,j) + \chi \\ \qquad \text{if } D_{\text{PGMV}}(i,j)^2 < \left(\dfrac{1-\chi}{\beta}\right)^2, \\ 1 \qquad \text{otherwise,} \end{cases} \tag{7}$$

where $(\min(D(i,j)) - \alpha)$ in (6) is subtracted from $D(i,j)$ because the magnitude of $D(i,j)$ can vary significantly according to the characteristics of an image. For example, the value of $D(i,j)$ is large when an image includes blurs and noises which may cause mismatches between successive frames. On the other hand, the value of $D(i,j)$ may be small for a clean image. The subtraction of $(\min(D(i,j)) - \alpha)$ reduces these differences in the amount of $D(i,j)$ between different types of images. The value of α is chosen as 10 by experiments. In (7), $g(D_{\text{PGMV}}(i,j))$, the parameters β and χ are chosen as 0.3 and 0.47, respectively, obtained from experimental results.

Let a new function $h(i,j)$ be defined as the product of $f(D(i,j))$ and $g(D_{\text{PGMV}}(i,j))$ as

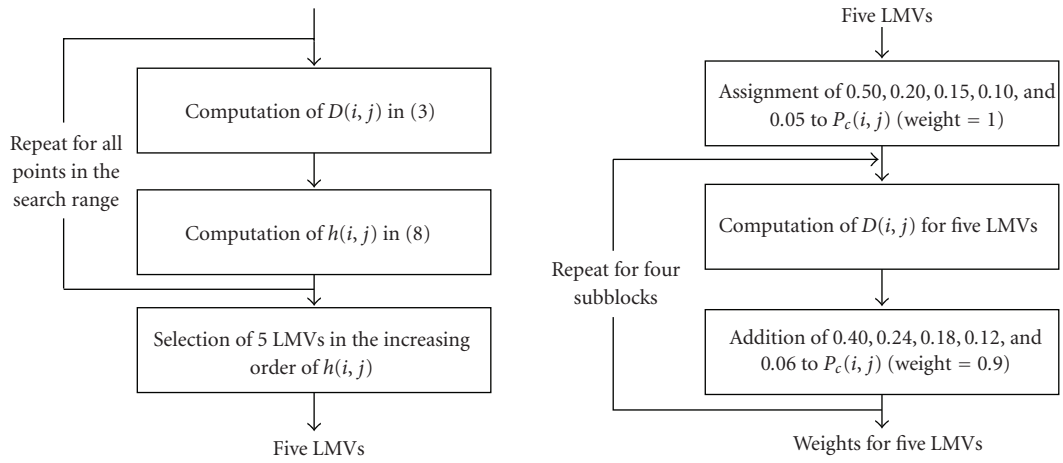
$$h(i,j) = f(D(i,j)) \times g(D_{\text{PGMV}}(i,j)). \tag{8}$$

Then, the probability increases as the value of $h(i,j)$ decreases. Five vectors with the smallest five values of $h(i,j)$ are selected as LMV candidates for each subblock. For these five



(a) Subimage (b) Subblock s_m (c) Subblocks s_1, s_2, s_3, and s_4

Figure 3: The method for searching LMVs from the subimage.



(a) Flow for LMV derivation (b) Flow for weight derivation

Figure 4: Computation flow of LMV and weight derivations.

vectors, its probabilities to become the GMV are assigned to be 0.50, 0.20, 0.15, 0.10, 0.05 for the corresponding LMVs in the increasing order of $h(i, j)$. This selection of probability values is very simple although these values may not be the optimal one. However, the simple derivation avoids the increase of the computation load by the probabilistic approach while this approximate probability values still give good results as shown in Section 4.

2.4. Subblock selection

The subimages for LMV derivation should be taken from a background region. If the whole or one part of a subimage is included in a foreground region moving in a different direction from the global motion, the MV obtained in such subimage should not be considered for the derivation of the GMV. This section proposes an algorithm to detect whether a part of a subimage is included in a foreground region or not. Based on the detection, the probabilities of candidate LMVs are adjusted accordingly.

Figure 3(a) shows a subimage S^i whose size is denoted by $W \times H$. Figure 3(b) shows a subblock s_m of size $w \times h$ located in the center of S^i. The size of a subblock $w \times h$ is equal to $0.5W \times 0.5H$. For the derivation of five candidate LMVs as explained in Sections 2.2 and 2.3, the subblock s_m is used instead of the entire subimage S^i. For all the points in the search space, $h(i, j)$ of (8) is calculated. Then, the five positions with the minimum $h(i, j)$ are selected as the five LMVs. The probabilities are also assigned to the five LMVs as explained in the last paragraph in Section 2.3. By using the small subblock s_m instead of the entire subimage S^i, the computational load is reduced. However, the accuracy of the estimated LMVs may also be reduced.

To compensate the reduction of the accuracy, the proposed algorithm uses additional four subblocks as shown in Figure 3(c). The size of these subblocks is also $0.5W \times 0.5H$. For these subblocks, $D(i, j)$ is calculated for the five LMVs derived with s_m. Note that this $D(i, j)$ calculation for each subblock is performed for only five points, that is, no searching operations are performed for these four additional subblocks. Thus, the increase of the computational load for these additional four blocks is negligible when compared with the expensive search operations performed for subblock s_m to derive the five LMVs.

The probabilities of the five LMVs initially assigned with s_m are adjusted based on the value of $D(i, j)$ calculated with s_1, s_2, s_3, and s_4. If this value is small for a given LMV, its probability should be increased. If not, the probability should be decreased. The values of 0.40, 0.24, 0.18, 0.12 and 0.06, in the increasing order of $D(i, j)$, are assigned, respectively, to the probabilities of the five LMVs. These probabilities are assigned for four subblocks s_1, s_2, s_3, and s_4. As a result, five probabilities are assigned to each LMV: one probability assigned with s_m as discussed in Section 2.3 and four probabilities with s_1, s_2, s_3, and s_4, respectively, as discussed in this subsection.

The five probabilities for each LMV are added to make a single value, called a weight. In this addition, the probability obtained with s_m is multiplied by 1 whereas the probabilities obtained with s_1, s_2, s_3, and s_4 are multiplied by 0.9. A bigger weight is given to the probability with s_m because the probability with s_m is obtained by full-search subblock matching and therefore it is more accurate than those with s_1, s_2, s_3, and s_4. Note that each subblock s_1, s_2, s_3, or s_4 requires only five calculations of $D(i, j)$ for the five candidate LMVs obtained with s_m, respectively. Therefore, the additional computational load for this calculation is small compared to the derivation of the five candidate LMVs which require $h(i, j)$ calculations for an entire search range.

This addition based on the amount of $D(i, j)$ for four subblocks contribute to the detection of a subimage in a foreground region. If $D(i, j)$ is large, this subimage may be in a foreground region so that a small value is added to the probability. On the other hand, $D(i, j)$ is relatively small if the subblock is included in a background region.

Figure 4 shows the flowchart that summarizes the computing process for the derivation of five candidate LMVs and their weights. Figure 4(a) shows the process computing five LMVs for subblock s_m. First, $D(i, j)$ of (3) is computed for all vectors in a search range. Second, $h(i, j)$ of (8) is computed to select the LMV with a small $D(i, j)$ value and close to the previous GMV. Then, five LMVs are selected based on the values of $h(i, j)$. Figure 4(b) shows the process for the derivation of weights for the five LMVs. First, the GMV probabilities of five LMVs are assigned to 0.50, 0.20, 0.15, 0.10, and 0.05, respectively, in the increasing order of the $h(i, j)$ value. This $h(i, j)$ is derived with the subblock s_m as shown in Figure 3. Second, $D(i, j)$ values are calculated for five LMVs and then the values of 0.40, 0.24, 0.18, 0.12, and 0.06 are assigned, respectively, to the GMV probabilities in the increasing order of the $D(i, j)$ value. These derivationsare repeated four times for four subblocks s_1, s_2, s_3, and s_4 of Figure 3(c). Finally, the five probabilities designated for each LMV are added to be the final weight of the LMV. In this addition, the probabilities assigned with subblock s_m are weighted by 1, and those with s_1, s_2, s_3, and s_4 are weighted by 0.9.

3. PROBABILISTIC GMV ESTIMATION

The LMVs obtained from the subimages are used for the derivation of the GMV. The five LMVs obtained from each subimage are denoted by $\mathrm{LMV}[0], \mathrm{LMV}[1], \ldots, \mathrm{LMV}[4]$, and their weights are denoted by $w[0], w[1], \ldots, w[4]$, respectively. The LMVs are sorted in the decreasing order of their weight values, that is, the value of $w[0]$ is the largest and the value of $w[4]$ is the smallest among all weights. When four subimages as shown in Figure 1(b) are used for the derivation of LMVs, the first step for GMV derivation is to select two subimages among the four subimages. This step contributes to the elimination of subimages that are included in a foreground region and generate inappropriate LMVs. The selection step consists of two substeps. The first substep is to select the first subimage while the second substep determines the other subimage.

The first subimage is selected as follows. The subimage with the largest weight of LMV[0] is chosen first among the four subimages. If two subimages have the same largest weight, the subimage with the third largest weight of LMV[0] is chosen and the Manhattan distances between the third LMV[0] and the LMV[0]s with the two largest weights are compared. Then, the subimage with the smaller Manhattan distance is chosen. If these two Manhattan distances are the same, any subimage is selected. For example, assume that LMV[0] of four subimages are (1,1), (2,3), (3,4), and (4,5) and that their respective weights are 1.50, 1.50, 1.45, and 1.40. Note that (1,1) and (2,3) have the same largest weight of 1.50. As the two vectors have the same weight, it is necessary to check the Manhattan distance between these vectors and the LMV[0] with the third largest weight. Note that the vector with the third largest weight is (3,4). The Manhattan distance between (1,1) and (3,4) is 5 while that between (2,3) and (3,4) is 2. Thus, the vector (2,3) is finally chosen and therefore the corresponding subimage is chosen as the first subimage.

The second substep for the selection of the second subimage is as follows. For the selection of the second subimage, the Manhattan distances between LMV[0] of the first subimage and LMV[0]s of the remaining subimages are derived. Then, the subimage with the smallest Manhattan distance is chosen for the second subimage. If these Manhattan distances are same, the subimage with the largest weight of LMV[0] is selected. If these weights are also the same, any subimage among them is selected. Consider again the example discussed above. Recall that vector (2,3) is chosen for the first subimage. From vector (2,3), the Manhattan distances of the other vectors, (1,1), (3,4), and (4,5) are 3, 2, and 4, respectively. Thus, the subimage with vector (3,4) is chosen for the second subimage.

In case that the Manhattan distance between PGMV and LMV[0] of a subimage is larger than 16, the corresponding subimage is not chosen for the derivation of the GMV. However, if the Manhattan distance between PGMV and LMV[0] is larger than 16 for all subimages, the two LMVs with the smallest Manhattan distance are chosen. The threshold value of 16 is chosen based on experiments. This constraint for the selection of the subimages is imposed because the GMV, in general, is not changed suddenly from PGMV [4]. Consider the above example again. Suppose that PGMV is (5,5). Then, the Manhattan distances between PGMV and the selected two vectors (2,3) and (3,4) are 5 and 3, respectively. As these distances are less than 16, the corresponding two subimages are selected for the next step (the derivation of GMV from LMVs). Suppose that PGMV is (11,11). Then, the Manhattan distance between (2,3) and (11,11) is 17. Thus, the corresponding subimage is not selected for the next step. On the other hand, the Manhattan distance between (3,4) and (11,11) is 15 which is less than 16. Thus, this subimage is chosen. The other vector (4,5) is also chosen because its Manhattan distance from PGMV is less than 16. Thus, with PGMV of (11,11), the two subimages corresponding to (3,4) and (4,5) are chosen finally.

Once two subimages are chosen, the GMV is derived from LMVs obtained from the two subimages. The algorithm

consists of five steps. If the condition of each step is satisfied, the algorithm stops in that step and does not move to the next step. The notations used in this algorithm are explained first and then the algorithm is presented next. The five LMVs for the first subimage are denoted by $\text{LMV}^L[0], \text{LMV}^L[1], \ldots, \text{LMV}^L[4]$. and their weights are denoted by $w^L[0], w^L[1], \ldots, w^L[4]$, respectively. The five LMVs for the second subimage are denoted by $\text{LMV}^R[0], \text{LMV}^R[1], \ldots, \text{LMV}^R[4]$. and their respective weights are denoted by $w^R[0], w^R[1], \ldots, w^R[4]$. The horizontal and vertical coordinates of $\text{LMV}^L[i]$ are denoted by $\text{LMV}^L_x[i]$ and $\text{LMV}^L_y[i]$, respectively. Similarly, the horizontal and vertical coordinates of $\text{LMV}^R[i]$ are denoted by $\text{LMV}^R_x[i]$ and $\text{LMV}^R_y[i]$, respectively.

Step 1. When there exists an $\text{LMV}^L[i]$ (or $\text{LMV}^R[i]$) that is the same as $\text{LMV}^R[0]$ (or $\text{LMV}^L[0]$), it is chosen as the GMV candidate. If the Manhattan distance between this GMV candidate and the PGMV is less than or equal to 16, the GMV candidate is chosen as the final GMV.

Step 2. If the value of $w^L[0]$ is greater than or equal to 1.49, and $w^L[0]$-$w^R[0]$ is greater than or equal to 0.25, $\text{LMV}^L[0]$ is chosen to be the GMV candidate. Similarly, if the value of $w^R[0]$ is greater than or equal to 1.49 and $w^R[0]$-$w^L[0]$ is greater than or equal to 0.25, $\text{LMV}^R[0]$ is chosen to be the GMV candidate. If the Manhattan distance between the GMV candidate and the PGMV is less than or equal to 16, the candidate is chosen as the final GMV.

Step 3. The Manhattan distance between $\text{LMV}^L[0]$ and $\text{LMV}^R[0]$ is computed. If this value is greater than or equal to 9, go to step 4. Otherwise, go to step 5.

Step 4. The LMV with the smallest Manhattan distance with the PGMV among the 10 LMVs is chosen to be the GMV.

Step 5. The average of $\text{LMV}^L[0]$ and $\text{LMV}^R[0]$ is assigned to be the final GMV.

steps 1 and 2 rectify the case when one subimage may be in a foreground region and provide wrong LMVs. In this case, the LMVs obtained from a wrong subimage are ignored and the GMV is chosen from one subimage. In step 2, the threshold 1.49 is obtained from experimental results as this value is the largest value of $w^L[0]$ (or $w^R[0]$) used for step 2 that does not decrease the accuracy of the proposed algorithm. Note that 1.49 represents 77% of the maximum value of a weight ($0.5 \times 1 + 0.4 \times 0.9 \times 4 = 1.94$) and the threshold 0.25 represents 13% of the maximum value. For example, suppose that the algorithm is at step 1 and both $\text{LMV}^L[1]$ and $\text{LMV}^R[0]$ are (3,1). Then, $\text{LMV}^L[1]$ and $\text{LMV}^R[0]$ are same, so that the final GMV candidate is chosen as (3,1). On the other hand, suppose that the algorithm is at step 2 and $\text{LMV}^L[0]$, $\text{LMV}^R[0]$, $w^L[0]$, and $w^R[0]$ are (−10,6), (7,−2), 1.28, and 1.60, respectively. Then, $w^R[0]$ is greater than 1.49, and $w^L[0]$-$w^R[0]$ is greater than 0.25. So, $\text{LMV}^R[0]$ is chosen to be the GMV candidate. When the algorithm passes step 3 and moves to step 4, the LMVs from both subimages are used for the derivation of the GMV. The general method is to determine the GMV as the average of $\text{LMV}^L[0]$ and $\text{LMV}^R[0]$ as in step 5. However, when the Manhattan distance between $\text{LMV}^L[0]$ and $\text{LMV}^R[0]$ is larger than 9, the compensation may deteriorate the image quality. This is due to the case when one of $\text{LMV}^L[0]$ and $\text{LMV}^R[0]$ is the correct GMV instead of the average of the two LMVs. Therefore, step 3 checks the Manhattan distance between these two LMV[0]s and performs different actions based on the Manhattan distance. Based on the experiment performed with various Manhattan distances, the threshold is chosen as 9. For example, suppose that the algorithm is at step 3 and $\text{LMV}^L[0]$ and $\text{LMV}^R[0]$ are (2,5) and (4,3). Then, the Manhattan distance between these two vectors are 4. Then, the next step is step 5 in which the final GMV is chosen as the average of these two vectors that is (3,4). For another example, $\text{LMV}^L[0]$ and $\text{LMV}^R[0]$ are (−5,1) and (2,4). Then, the algorithm goes to step 4 in which the final GMV is chosen as the LMV which is closest to the PGMV.

Figure 5: Sample frame of the foreman image sequence.

With the final GMV obtained by the aforementioned algorithm, a global motion correction is performed by the MVI method as described in [4] for camera fluctuation.

4. EXPERIMENTAL RESULTS

Experimental results of the proposed algorithm are presented and compared with existing algorithms in this section. The foreman video sequence with the CIF size at 15 frames per second is used. The foreman sequence includes clockwise rotations as well as translational jitters. Figure 5 shows a sample frame of the foreman sequence. As shown in this sample, the image includes a single human as the only foreground object. Therefore, two subimages are used to obtain LMVs. The squares with solid lines in the upper corners of Figure 5 represent the subimages in the current image used for obtaining LMVs. The squares with dotted lines represent the search range in the previous image. The sizes of these squares are 64×64 and 96×96, respectively.

The performance evaluation uses the measure of root mean square error (RMSE)

$$\text{RMSE} = \frac{1}{N}\sqrt{\sum_{k=1}^{N}(x_k - \hat{x}_k)^2 + (y_k - \hat{y}_k)^2}, \quad (9)$$

where $(\hat{x}_k, \hat{y}_k)$ is the optimal GMV and (x_k, y_k) is the GMV obtained by the proposed algorithm as well as other previous algorithms to be compared. The optimal GMV is obtained by FS-BM based on the MSE criterion. The number of frames is denoted by N which is 150 in the foreman sequence used in this experiment.

The RMSE values obtained for various DIS algorithms are shown in the Table 1. The SM-MAD represents the DIS algorithm with the matching criterion of MAD. This algorithm uses two subimages as shown in Figure 5 to perform motion estimation and obtain two LMVs, one for each of these two subimages. Then, the GMV is derived by averaging these LMVs. FS-FM is not compared in this experiment because the foreman sequence has a large foreground object with a large movement, and therefore, the frame matching using an entire frame does not lead to an accurate motion estimation. The SM-EPM algorithm performs matching of edge regions in the subimage [6]. GCBPM-W and GCBPM-4 represent gray coded bit-plane matching algorithms [3, 4]. The GCBPM-W algorithm uses matching bitplanes weighted by their respective levels of significance and the GCBPM-4 algorithm uses matching only the 4th-order bit-plane. 1BT represents the method with one-bit-per-pixel transform [10]. Three phased-correlation-based motion estimation algorithms (PC-H, PC-2, and PC-W) are also compared. PC-H represents the algorithm choosing the LMV with the largest amplitude as the GMV [1]. In PC-2 algorithm, the average of the two LMVs with the largest two amplitudes is determined as the GMV [1]. In PC-W algorithm, the GMV is the weighted average of all LMVs based on their peak amplitude values [1]. The proposed algorithm is denoted by P-L2BT where P stands for the Probabilistic global motion estimation. SM-L2BT also uses the Laplacian transform to convert an image into two one-bit planes as described in Section 2.2. Then, SM-L2BT performs full subimage matching with these two planes as SM-MAD. As a result, SM-L2BT requires a lot more computation than P-L2BT. Table 1 shows that the proposed P-L2BT algorithm produces the least RMSE among all the compared algorithms. When compared with SM-MAD, P-L2BT reduces the RMSE by 27.7%.

Table 2 shows the contribution of each part of the proposed algorithm to the performance improvement. To this end, only certain parts of the proposed algorithm are performed and their results are compared. The first column of Table 2 shows which part of the algorithm is performed. The second and third columns show the RMSE and the number of execution cycles. The ARM Developer Suite 1.2 is used for measuring the clock cycle, and the target processor is ARM7TDMI processor. In the method represented by the second row, the distance of (3) is used for the matching criterion to obtain the LMV for each subimage. Then, the average of the two LMVs derived from the two subimages is chosen as the GMV. Note that the LMVs are derived with subblock s_m as shown in Figure 3. The method in the third row uses (8) instead of (3) as the matching criterion. As a result, RMSE is reduced by 6.35%, while the computation load is increased by only 0.07%. The method in the fourth row shows how much performance improvement is achieved by using five LMVs for each subimage together with the algorithms described in Section 3. Note that the method does not use the four subblocks as described in Section 2.4. In other words, it just uses the technique described in Section 2.3. The result shows that the RMSE is significantly reduced while the increase of the computational load is negligible. The final row shows the result of the proposed algorithm including all techniques described from Sections 2.1 to 3. RMSE is reduced by 61.27% while the computational load is increased by only 3.25%. Each process contributes to the significant improvement of RMSE while the increase of computational load is small. The small increase of the computational load is because distance calculations for many possible positions in the search range consume most of the computation.

Table 1: RMSE comparison of DIS algorithms experimented with the foreman sequence.

DIS algorithms	RMSE
SM-MAD	0.05201
SM-EPM	0.14237
GCBPM-W	0.11322
GCBPM-4	0.07688
1BT	0.14824
PC-H	0.12951
PC-2	0.08822
PC-W	0.09437
SM-L2BT	0.06766
P-L2BT	0.03756

Table 2: Contribution of each part of the P-L2BT algorithm with the foreman sequence.

Part of the L2BT algorithm	RMSE	Clock cycle
Laplacian 2-bit transform (L2BT)	0.22377	42,612,036
L2BT + PGMV weight	0.20955	42,641,911
L2BT + PGMV weight + 5 LMVs	0.09698	42,642,086
L2BT + PGMV weight + 5 LMVs + 4 subblocks (P-L2BT)	0.03756	44,029,002

The performance of the proposed algorithm is also experimented with the dancing boy sequence. The dancing boy sequence shows larger fluctuations than the foreman sequence. In addition, this sequence has more noises and blurs than the foreman sequence. Figure 6 shows an example frame of the QVGA (320×240) size. Since this is not a single human

Figure 6: Sample frame of the dancing boy image sequence.

Table 3: RMSE comparison of DIS algorithms experimented with the dancing boy sequence.

DIS algorithms	RMSE
SM-MAD	0.03779
SM-EPM	0.25934
GCBPM-W	0.11395
GCBPM-4	0.09691
1BT	0.06993
PC-H	0.19067
PC-2	0.14368
PC-W	0.14094
SM-L2BT	0.04967
P-L2BT	0.03836

Table 4: Contribution of each part of the P-L2BT algorithm with the dancing boy sequence.

Part of the L2BT algorithm	RMSE	Clock cycle
Laplacian 2-bit transform (L2BT)	0.15939	88,878,765
L2BT + PGMV weight	0.13800	88,942,015
L2BT + PGMV weight + 5 LMVs	0.13450	88,942,253
L2BT + PGMV weight + 5 LMVs + 4 subblocks (P-L2BT)	0.03836	91,815,708

image, four subimages are used to obtain LMVs. The subimages with search ranges are shown as squares in Figure 6. The squares with solid lines represent the subimages of the current image and the squares with dotted lines represent search ranges in the previous image. The size of these subimages is the same as that of the foreman sequence.

Table 3 shows the RMSE values measured with various DIS algorithms. SM-MAD and P-L2BT algorithms are the two best algorithms. When compared with the foreman sequence, the SM-MAD algorithm is slightly improved while the P-L2BT algorithm results in almost the same RMSE.

For the other algorithms, the RMSE values increase badly except the GCBPM-W, GCBPM-4, and 1BT algorithms. This is because these algorithms are sensitive to noises or blurs.

Table 4 shows the contributions of various techniques used by the proposed DIS algorithm. Each row represents the same part as Table 2. The third row shows that the use of (8) instead of (3) reduces RMSE by about 13.42% while the computational load is increased only by 0.07% when compared with the second row. The fourth row shows that the use of five LMV candidates with the decision algorithm in Section 3 reduces RMSE by 2.54% while the increase of the computational load is negligible. The bottom row shows that the use of four subblocks reduces RMSE by 71.48% while the computational load is only increased by 3.23%. The measurement with the dancing boy sequence also shows that the use of four subblocks gives the largest improvement.

Table 5: RMSE comparison of the probabilistic version of 1BT, SM-EPM, GCBPM-4, first derivatives, and L2BT.

	Foreman	Dancing boy	Motorcycle
P-1BT	0.08186	0.03223	0.08836
P-SM-EPM	0.10698	0.09868	0.25420
P-GCBPM-4	0.05080	0.06773	0.19413
P-first derivatives	0.04827	0.05659	0.16651
P-L2BT	0.03756	0.03836	0.08707

Table 5 compares the RMSE of five correlation matching methods, 1BT, SM-EPM, GCBPM-4, first derivatives, and P-L2BT to which the probabilistic approach is applied, so that all the probabilistic techniques used for P-L2BT are also implemented for the other algorithms. These techniques include the use of 5 LMVs with probabilities, the use of (8) to give a weight to PGMV, the probabilistic adjustment with 4 subblocks of Figure 3(c), and the GMV derivation algorithm proposed in Section 3. For the first derivatives method, two spaces similar to L^+ and L^- spaces are used. Therefore, two bits are required to represent one pixel. Three test sequences, foreman, dancing boy, and motorcycle sequences are used for evaluation. In the table, the prefix P- represents the probabilistic version of the previous algorithms. As shown in the table, 1BT, SM-EPM and GCBPM-4 algorithms are significantly improved by the probabilistic approach when compared with the results shown in Tables 1 and 3. The P-L2BT algorithm achieves the smallest RMSE for foreman and motorcycle sequences. For dancing boy sequence, the RMSE of P-L2BT is slightly larger than that of 1BT but the difference is negligible.

In general, the search operation for motion estimation requires about 90% of the total computation of DIS [16]. Therefore, the number of operations required for the motion estimation is compared. Table 6 shows the comparison of the computational complexity of DIS algorithms. Among the algorithms presented in Tables 1 and 3, SM-MAD and three phase-correlated-based motion estimation algorithms (PC-H, PC-2, and PC-W) are excluded. SM-MAD requires the sum of the absolute pixel-by-pixel difference (SAD) operations between the target block and the block at a matching location on the reference frame which are more complex than the Boolean operations required for the other algorithms. The three phase-correlated algorithms employ the 2D Fourier transform that also requires significantly larger complexity than the others [10]. In Table 6, W and H represent the width and the height of a sub-image and$r$ denotes the search range as given in (3). SM-EPM, GCBPM-4, 1BT

Table 6: The number of operations required by motion estimation.

Algorithms	Boolean Exclusive-Or	Boolean-Or
SM-EPM, GCBPM-4, 1BT	$W \times H \times (2r+1)^2$	0
GCBPM-W	$W \times H \times 9\log_2(2r+1)$	0
SM-L2BT	$W \times H \times 2(2r+1)^2$	$W \times H \times (2r+1)^2$
P-SM-EPM, P-GCBPM-4, P-1BT	$\frac{W \times H}{4} \times (2r+1)^2 + 20 \times W \times H$	0
P-L2BT	$\frac{W \times H}{2} \times (2r+1)^2 + 40 \times W \times H$	$\frac{W \times H}{4} \times (2r+1)^2 + 20 \times W \times H$

Table 7: Improvement of the compression efficiency by the proposed DIS algorithm.

Frame	Foreman		Dancing boy	
	PSNRΔ [dB]	bitΔ [%]	PSNRΔ [dB]	Δ bit [%]
1–10	0.14	12.69	−0.17	1.48
11–20	0.05	2.88	−0.05	3.01
21–30	−0.03	0.72	−0.03	−0.74
31–40	−0.13	−3.79	0.05	1.67
41–50	−0.06	−1.76	0.11	2.44
51–60	0.01	4.21	−0.06	1.07
61–70	0.03	4.69	−0.02	1.76
71–80	0.26	6.28	−0.08	8.29
Average	0.03	3.24	−0.03	2.37

and GCBPM-W require one Boolean Exclusive-Or operation per each point in the search range. SM-EPM, GCBPM-4, 1BT employs the full search algorithm while GCBMP-W employs the three step search algorithm for motion estimation. Thus, the GCBPM-W requires the smallest number of operations. For L2BT, the number of the operations to derive $D(i, j)$ in (3) is used for this comparison. The precise comparison needs to include the number of operations to derive $h(i, j)$ in (8). However, the difference between the numbers for $D(i, j)$ and $h(i, j)$ is not large because the number for $D(i, j)$ is much larger than that for $h(i, j)$. The derivation of $D(i, j)$ requires two Boolean Exclusive-Or and one Boolean-Or operations in (3). For SM-L2BT, $W \times H \times 2(2r+1)^2$ and $W \times H \times 2(2r+1)^2$ operations are required for Boolean Exclusive-Or and Boolean-Or, respectively. On the other hand, the numbers of search points for P-SM-EPM, P-GCBPM-4, P-1BT (the probabilistic versions of SM-EPM, GCBPM-4, 1BT) and P-L2BT are reduced because the NNMP operations are performed for the subblock s_m instead of the entire sub-image S^i. For P-SM-EPM, P-GCBPM-4, P-1BT and P-L2BT, NNMP operations are required to derive $D(i, j)$ for each of the $(W \times H)/4$ pixels. The P-SM-EPM, P-GCBPM-4, P-1BT use 1 bit per pixel, and the term $20 \times W \times H$ for Boolean Exclusive-Or is needed to derive $D(i, j)$ for the four subblocks s_1, s_2, s_3, and s_4. For P-L2BT using 2 bits per pixel, the terms $40 \times W \times H$ for Boolean Exclusive-Or and $20 \times W \times H$ for Boolean-Or are required to derive $D(i, j)$ for four subblocks s_1, s_2, s_3, and S_4. Thus, P-L2BT requires about 3 times larger amount of computation than P-SM-EPM, P-GCBPM-4 or P-1BT.

If a video sequence is compensated by DIS, it may be compressed more efficiently than the original video sequence. This is because the compensation of global motion may reduce the prediction errors between the current and previous frames. To evaluate the improvement of the compression efficiency by DIS, the video sequences before and after the execution of the P-L2BT algorithm are used as the inputs to the H.264 encoder and the compression ratios for these two sequences are compared. The version 7.3 of JM reference software for the H.264/AVC baseline profile encoder is used for experiments. Table 7 shows the improvement of the compression ratio (denoted by Δbit [%]) and the improvement of PSNR (denoted by ΔPSNR [dB]) by the proposed DIS algorithm. The first frame is encoded as an I-frame and all the remaining frames are encoded as a P-frame, and QP value is set to 25. Note that the compression efficiency is not improved for an I-frame which does not use temporal reference frames. Eighty frames of both foreman and dancing boy sequences are used and the compression efficiencies are averaged for every ten frames. For the first ten frames of the foreman sequence, the compression ratio improvement of 12.69% is achieved. This is because these frames include large fluctuations. For the dancing boy sequence, the biggest compression ratio improvement of 8.29% is achieved for 71–80 frames which also include large fluctuations. On average, the foreman sequence improves the PSNR and the compression ratio by 0.03 dB and 3.24% per a P-frame, and the dancing boy sequence achieves the compression ratio improvement of 2.37% per a P-frame while the PSNR is decreased by 0.03 dB per a P-frame.

5. CONCLUSION

In the proposed algorithm, five LMVs with their respective probabilities to be the GMV are assigned for each subimage and then the GMV is derived from the LMVs. This probabilistic approach is pretty effective while the additional computation load is not heavy compared to the LMV derivation which requires expensive matching operations over a large search space. Therefore, the proposed DIS algorithm successfully reduces the computational burden while maintaining the quality of the stabilized video. As a result, the algorithm can be adopted in handheld devices for real-time video capturing applications.

Many parameters used in this paper are also determined by experimental results. Based on the experimental results,

the proposed probability model is designed to be simple, but it is not the optimal one. For example, the proposed probability model used in (8) is not the optimal one and the probability values chosen in the last paragraph of Section 2.3 are also chosen to simplify the complexity of calculation. Thus, the contribution of this paper is the proposal of the probabilistic approach to derive GMV, but the probabilistic model is not the optimal one and further research is necessary to search the optimal probability model.

In [16], DIS is combined with a video encoder and the complexity of DIS is reduced by using the motion estimation information generated by the video encoder. The idea of the above paper is promising, and its application to the L2BT DIS is a good future research topic.

ACKNOWLEDGMENT

This work was supported by "system IC2010" project of Korea Ministry of Commerce, Industry, and Energy.

REFERENCES

[1] S. Ertürk, "Digital image stabilization with sub-image phase correlation based global motion estimation," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 1320–1325, 2003.

[2] F. Vella, A. Castorina, M. Mancuso, and G. Messina, "Digital image stabilization by adaptive block motion vectors filtering," *IEEE Transactions on Consumer Electronics*, vol. 48, no. 3, pp. 796–801, 2002.

[3] S.-J. Ko, S.-H. Lee, and K.-H. Lee, "Digital image stabilizing algorithms based on bit-plane matching," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 617–622, 1998.

[4] S.-J. Ko, S.-H. Lee, S.-W. Jeon, and E.-S. Kang, "Fast digital image stabilizer based on gray-coded bit-plane matching," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 3, pp. 598–603, 1999.

[5] K. Uomori, A. Morimura, and H. Ishii, "Electronic image stabilization system for video cameras and VCRs," *Journal of the Society of Motion Picture and Television Engineers*, vol. 101, no. 2, pp. 66–75, 1992.

[6] J. K. Paik, Y. C. Park, and D. W. Kim, "An adaptive motion decision system for digital image stabilizer based on edge pattern matching," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 3, pp. 607–616, 1992.

[7] F. Vella, A. Castorina, M. Mancuso, and G. Messina, "Robust digital image stabilization algorithm using block motion vectors," in *Proceedings of IEEE International Conference on Consumer Electronics (ICCE '02)*, pp. 234–235, Los Angeles, Calif, USA, June 2002.

[8] K.-S. Choi, J.-S. Lee, J.-W. Kim, S.-H. Lee, and S.-J. Ko, "An efficient digital image stabilizing technique for mobile video communications," in *Proceedings of IEEE International Conference on Consumer Electronics (ICCE '00)*, pp. 246–247, Los Angeles, Calif, USA, June 2000.

[9] Y. T. Tse and R. L. Baker, "Global zoom/pan estimation and compensation for video compression," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '91)*, vol. 4, pp. 2725–2728, Toronto, Ontario, Canada, April 1991.

[10] A. A. Yeni and S. Ertürk, "Fast digital image stabilization using one bit transform based sub-image motion estimation," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 3, pp. 917–921, 2005.

[11] G. Qiu and C. Hou, "A new fast algorithm for the estimation of block motion vectors," in *Proceedings of the 3rd International Conference on Signal Processing (ICSP '96)*, vol. 2, pp. 1233–1236, Beijing, China, October 1996.

[12] T. Ha, S. Lee, and J. Kim, "Motion compensated frame interpolation by new block-based motion estimation algorithm," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 2, pp. 752–759, 2004.

[13] S. Ertürk and T. J. Dennis, "Image sequence stabilization based on DFT filtering," *Proceedings of IEE on Image Vision and Signal Processing*, vol. 127, no. 2, pp. 95–102, 2000.

[14] S. Ertürk, "Real-time digital image stabilization using Kalman filters," *Real-Time Imaging*, vol. 8, no. 4, pp. 317–328, 2002.

[15] L. Hill and T. Vlachos, "Motion measurement using shape adaptive phase correlation," *Electronics Letters*, vol. 37, no. 25, pp. 1512–1513, 2001.

[16] H. H. Chen, C.-K. Liang, Y.-C. Peng, and H.-A. Chang, "Integration of digital stabilizer with video codec for digital video cameras," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 7, pp. 801–813, 2007.

[17] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Prentice Hall, Englewood Cliffs, NJ, USA, 1992.

[18] A. Ertürk and S. Ertürk, "Two-bit transform for binary block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 7, pp. 938–946, 2005.

[19] N. J. Kim, S. Ertürk, and H. J. Lee, "Two-bit transform based block motion estimation using second derivatives," in *Proceedings of IEEE International Conference on Multimedia & Expo (ICME '07)*, pp. 1615–1618, Beijing, China, July 2007.

[20] S. J. Park, N. J. Kim, and H. J. Lee, "Binary integer motion estimation and SAD-merge based fractional motion estimation for H.264/AVC," in *Proceedings of International Soc Design Conference (ISOCC '07)*, pp. 329–332, Seoul, Korea, October 2007.