International Journal of the Physical Sciences Vol. 6(7), pp. 1665-1672, 4 April, 2011
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS11.246
ISSN 1992 - 1950 ©2011 Academic Journals

Full Length Research Paper

Real-time algorithmic design for silent pass lip reading authentication system

Khaled Alghathbar[1, 2]

[1]Center of Excellence in Information Assurance, King Saud University, Riyadh, Kingdom of Saudi Arabia.
[2]College of Computer and Information Sciences, King Saud University, Riyadh, Kingdom of Saudi Arabia. E-mail: kalghathbar@ksu.edu.sa.

Accepted 22 March, 2011

This paper proposes a new block-matching motion estimation algorithm for a lip reading user authentication system. The method described in this paper represents a sub-system of the Silent Pass project. Silent Pass is a lip reading password entry system and person authentication project for security applications. The aim is to provide a complete solution for secured access to ATMs and internet services in multi-media environments. The proposed block matching algorithm is an optimal algorithm that mimics the actions of an exhaustive full search block matching algorithm (FSBM). The proposed algorithm reduces the computational load of FSBM by pruning non-candidate blocks from the search window using approximate SAD (sum of absolute differences) values. This computational reduction leads to enhanced performance in terms of speed and reduced computational load for motion estimation. Lip reading authentication is a real-time application that makes FSBM an unrealistic choice because of its low speed. The proposed algorithm provides the same result as the FSBM algorithm at a high speed.

Key words: Human computer interaction (HCI), full search block matching (FSBM), pruning full search block matching (PFSBM), VLSI architecture.

INTRODUCTION

Recently, lip-reading has become a hot topic for human computer interaction (HCI) and audio-visual speech recognition (AVSR). Lip-reading systems can be utilized in many applications such as assistance for the hearing impaired and in noisy environments where speech is highly unrecognizable. A lip-reading system is essentially a signal processing system, in which feature extraction plays a crucial role. Currently, various visual features have been proposed in the literature and motion estimation is used for extracting such features. Motion estimation is the central component used to represent and recognize speech from lip reading, and optimal motion estimation is a computationally expensive operation. Motion analysis techniques are used to generate the motion vectors that are used for motion estimation. The full-search block-matching motion estimation (FSBM) algorithm provides the optimum solution by an exhaustive evaluation of all of the blocks in the search window for the best solution. FSBM is computationally expensive, especially in the power consumption of the device. Up until now, several cost effective techniques at the algorithmic level have been reported in the literature (Cheung and Po, 2005; Gao et al., 2000; Brnig and Niehsen, 2001; Ahn et al., 2004). The method described in this paper represents a sub-system of the Silent Pass project (Mahmoud et al., 2009). Silent pass is a lip reading password entry system and personal authentication project for security applications. The aim is to provide a complete solution for secured access to ATM machines and internet services in multi-media environments. The methodology is expected to improve the performance of authentication systems and reduce the acceptance of impostors and shoulder-surfing threats, which are very common in the traditional PIN

code-based authentication systems.

In this paper, we propose an enhancement to the existing FSBM algorithm, which reduces the algorithmic complexity, as well as the power consumption, while retaining the optimal solution. Our approach is based on using approximate values for the sum of the absolute differences to prune non-candidate blocks at a preliminary stage. This will drastically reduce the computational load. The proposed algorithm produces the optimal solution in less time than FSBM. Furthermore, low-power VLSI architecture for the proposed FSBM algorithm is also presented in this paper.

The rest of the paper is organized as follows: First is a presentation of the background and related work. Next is a discussion of the new Pruning FSBM algorithm (PFSBM); followed by the proposed low-power VLSI architecture. Experimental results are then discussed; and finally, presentation of the findings of this paper concludes this study.

Background and related work

The major aspects of motion estimation techniques are:

(i) Computational complexity.
(ii) Good visual quality in terms of true motion representation.
(iii) High compression ratio.

The computational complexity of a motion estimation technique can be determined by three factors:

(i) The search algorithm: decides the overall computational complexity and motion estimation accuracy.
(ii) Search area: the span of the search window used to find the best match.
(iii) Cost function: the different cost metrics used to find the best match.

The exhaustive search algorithm (ESA) compares the current block to all of the candidate blocks, and selects the motion vector corresponding to the candidate block, which yields the best criterion function value. The typical criteria used are the sum of absolute differences (SAD) and sum of squared differences (SSD). There are many fast algorithms for block motion estimation. For example, the three step search (TSS) reduces the computation by testing only the most promising motion vector candidates and ignoring the rest. This degrades the estimation result because it only retains the local optimal from the selected candidates. An alternative class of fast full search algorithms test all of the candidate vectors without degrading the result. They compute the SAD lower bound for the current vector, and compare it to the best SAD found so far. The successive elimination algorithm (SEA) (Li and Salari, 1995), Liu et al. (2008) uses an upper bound for a block sum difference as the criterion to eliminate impossible candidate blocks and reduce the motion estimation computation. The drawback of SEA is that the difference of block sums is not close enough to the true SAD. The SEA extended a multilevel successive elimination strategy (MSEA) (Gao et al., 2000) that provides tighter and tighter boundary values from the lowest level to the highest level. MSEA built an image pyramid structure for the current and reference blocks with $\log_2 n$ levels. SEA can be regarded as a special case of MSEA when only level zero is used to eliminate impossible candidates. The extended SEA (ESEA) (Brnig et al., 2001) reuses the boundary value of the lowest level in the SAD calculation (Liu et al., 2008). A new algorithm called hierarchical diamond search (HDS) is proposed in (Urban et al., 2009) where HDS motion estimation is integrated in an AVC encoder. In (Ndili and Ogunfunmi, 2009), they presented an architecture based on a modified hybrid FME algorithm, where the main contributions are the hardware modifications to the hybrid FME algorithm. The architecture is based on the modified hybrid FME. The authors in Cheng and Dung (2005) present a novel power-aware motion estimation algorithm, called adaptive content-based subsample algorithm (ACSA), for battery-powered multimedia devices. When the battery status changes, the architecture adaptively performs graceful tradeoffs between power consumption and compression quality.

Full-search block-matching algorithm

The full-search block matching algorithm (FSBM) finds the best match for a reference block in the current frame within search area S of the previous frame. The criterion for the best match is the candidate block with the minimum amount of distortion when compared with the reference block. The measure used for calculating distortion is the sum of the absolute differences (SAD) in the intensity values between the two blocks. The SAD for a candidate block of size N × N at position (u, v) can be defined as:

$$SAD\ (u,v) = \sum_{i=1}^{N}\sum_{j=1}^{N} \left| u(i+u,\ j+v) - v(i,\ j) \right| \quad (1)$$

where v(i, j) and u(i+u, j+v) are the intensity values at position (i, j) of the reference block and (i+u, j+v) of the candidate block in search area S. The search area is formed by extending the reference block by search range w on each side, forming a search area of (2w+N)2 pixels. As a result, there are (2w+1) candidate blocks in both the horizontal and vertical directions, that is, a total of (2w+1)2 candidate blocks have to be searched corresponding to each reference block. The distortion value is computed for each candidate block and the minimum value, SAD min, is found. The block matching process generates a motion vector, (u, v) min, and the corresponding distortion value, SAD min.

The new FSBM algorithm based on the approximate-sad: PFSBM

The PFSBM algorithm

The proposed algorithm makes an integral approximation of the DC values of individual blocks. The DC value of block r is calculated as presented in Equation 2. The calculation of the new estimated DC value requires less power consumption compared to the computation of the conventional one. Two approximates will be calculated for the DC value, an upper

bound and a lower bound. The calculation of the upper bound of the DC of block u(DCU) is given in Equation 3. The calculation of the lower bound of the DC of block u(DCL) is given in Equation 4.

$$DC \quad (r) = \sum_{i=1}^{N} \sum_{j=1}^{N} r(i, j) \tag{2}$$

$$DC^{U}(r) = \sum_{i=1,step\,2}^{N} \sum_{j=1,step\,2}^{N} \left| r(i, j) \oplus r(i+1, j+1) \oplus (2^{N} - 1) \right| \tag{3}$$

$$DC^{L}(r) = \sum_{i=1,step\,2}^{N} \sum_{j=1,step\,2}^{N} \left| r(i, j) \oplus r(i+1, j+1) \oplus 0 \right| \tag{4}$$

where DCU indicates that all of the additions using r will be assumed to generate a "1" as a carry and DCL indicates that all of the additions using r will be assumed to generate a "0" as a carry.

The proposed algorithm (PFSBM)

The proposed algorithm is a block based motion estimation algorithm that utilizes a pruning technique more efficiently. In FSBMA, SAD is calculated for every candidate block. This leads to a complexity of almost N4. Since motion estimation is a part of a lip reading user authentication system, real time constraints might apply. Therefore, the high complexity of the motion estimation algorithm is of great concern, as it might conflict with the real time constraints. In the new proposed algorithm, the complexity of the algorithm is much lower than FSBMA. The elimination of candidate blocks without performing a full computation leads to less computation, resulting in large power savings with proven accuracy. This is because the algorithm eliminates blocks that will be eliminated by FSBMA with absolute certainties.

Calculation of SAD(u, v)

In this section, we will derive the approximate functions SADU(u, v) and SADL(u, v) for the first stage elimination algorithm. The intensity level of pixel (I) takes values between 0 and 2b where 'b' is the number of bits used to represent the resolution, which is usually 8-bits. The first stage utilizes the two approximate functions DCU and DCL, as indicated in Equations 5 and 6.

$$SAD^{U}(u, v) = DC^{U}(u) - DC^{L}(v) \tag{5}$$

$$SAD^{L}(u, v) = \min((DC^{L}(u) - DC^{L}(v), (DC^{L}(v) - DC^{L}(u)) \tag{6}$$

where DCL in Equation (6) is calculated without the absolute value calculations to ensure a minimum value, u is the reference block, and v is the candidate block. In lemma 1 and 2, we will prove that SADU(u, v) is an upper limit for SAD(u, v) and SADL(u, v) is a lower limit for SAD(u, v).

Lemma 1: SADU(u, v) is an upper limit of SAD(u, v).

Proof:

$$SAD^{U}(u, v) = DC^{U}(u) - DC^{L}(v)$$

$$= \sum_{i=1,step\,2}^{N} \sum_{j=1,step\,2}^{N} \left| u(i, j) \oplus u(i+1, j+1) \oplus (2^{N} - 1) \right| -$$

$$\sum_{i=1,step\,2}^{N} \sum_{j=1,step\,2}^{N} \left| v(i, j) \oplus v(i+1, j+1) \oplus 0 \right|$$

$$= \sum_{i=1,step2}^{N} \sum_{j=1,step2}^{N} \left| u(i, j) \oplus u(i+1, j+1) \oplus c_u(i, j) + (2^{N} - 1) - c_u(i, j)) \right|$$

$$- \sum_{i=1,step2}^{N} \sum_{j=1,step2}^{N} \left| v(i, j) \oplus v(i+1, j+1) \oplus c_v(i, j) - c_v(i, j) \right|$$

where cu (i, j) represents the carry bits generated by adding u(i, j) and u(i+1, j+1), and cv(i, j) represents the carry bits generated by adding v(i, j) and v(i+1, j+1).

$$= \sum_{i=1,step\,2}^{N} \sum_{j=1,step\,2}^{N} \left| u(i, j) + u(i+1, j+1) + (2^{N} - 1) - c_u(i, j)) \right|$$

$$- \sum_{i=1,step\,2}^{N} \sum_{j=1,step\,2}^{N} \left| v(i, j) + v(i+1, j+1) - c_v(i, j) \right|$$

$$SAD^{U}(u, v) = SAD(u, v) + \sum_{i=1}^{N} \sum_{j=1}^{N} \left| 2^{2N-1} + c_v(i, j) - c_u(i, j) \right| \tag{7}$$

The maximum value of cv(i, j) is 22N-1 and the maximum value for cu(i, j) is 22N-1. Therefore, the difference between them should be less than or equal to -22N-1, as shown in Equation 8.

$$2^{2N-1} + c_v(i, j) - c_u(i, j) \geq 1 \tag{8}$$

From Equations 7 and 8,

$$SAD^{U}(u, v) \geq SAD(u, v) \tag{9}$$

From Equation 9, we can conclude that SADU(u, v) is an upper limit of SAD(u, v).

Lemma 2: SADL(u, v) is a lower limit of SAD(u, v).

$$SAD^{L}(u, v) = \min((DC^{L}(u) - DC^{L}(v), (DC^{L}(v) - DC^{L}(u)),$$

where DCL is calculated without the absolute value calculations to ensure a minimum value.

$$SAD^{L(9)}(u, v) = \min((\sum_{i=1,step2}^{N} \sum_{j=1,step2}^{N} u(i, j) \oplus u(i+1, j+1) \oplus 0 -$$

$$v(i, j) \oplus v(i+1, j+1) \oplus 0), (\sum_{i=1,step2}^{N} \sum_{j=1,step2}^{N} v(i, j) \oplus v(i+1, j+1) \oplus 0 -$$

$$u(i, j) \oplus u(i+1, j+1) \oplus 0))) \tag{10}$$

Since SADL(u, v) is computed by ignoring the carry generated from the addition processes, it implies that it is a lower limit to SAD(u, v).

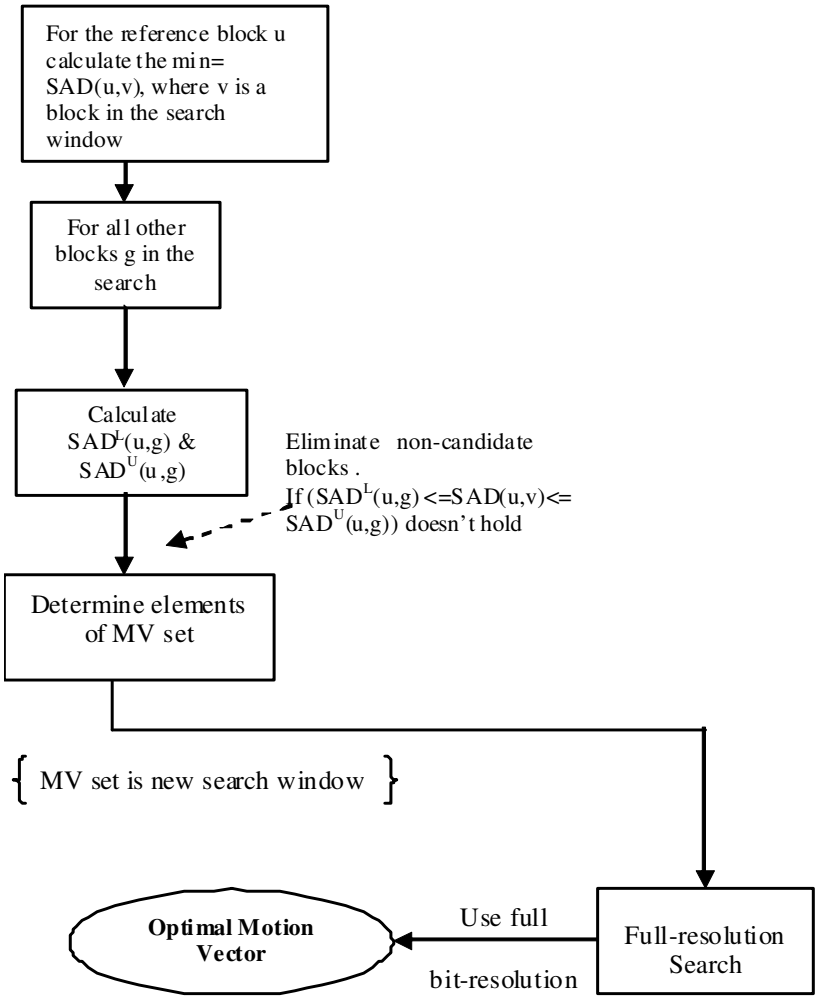

Figure 1. The proposed algorithm PFSBM for obtaining the possible motion vector set reference block.

Equations (9) and (10) can also be presented as follows:

$$SAD^{L}(u,v) \le SAD(u,v) \le SAD^{U}(u,v) \quad (11)$$

Two-stage motion estimation algorithm

The two-stage motion estimation algorithm uses interval-based matching in the first stage, and full-resolution matching in the second and final stage. The first stage uses the calculations of SADU and SADL. The calculation of the SAD between the reference block and one of the candidate blocks in the search window is performed. This value is assumed to be the minimum SAD. Then, the algorithm calculates both SADU and SADL for all of the other candidate blocks. If the minimum SAD falls between SADU and SADL for a candidate block, then this block will be contained in the MV set, which constitutes the new search window, otherwise the block will be eliminated with absolute certainty. This step reduces the number of candidate blocks in the search window. The first step results in a possible motion vector set MV. This set is further refined by applying the final stage, which determines the value of the optimal motion vector. The detailed algorithm for obtaining the possible motion vector set is presented below in Figure 1. An example showing the methodology is presented below.

Example

Reference block B has the dimensions 2 × 2 and the values of B = {1, 1, 3, 2} row wise.
If the search windows have candidate blocks B1, B2, B3, B4, and B5, as follows:

Table 1. DCU and DCL.

Block	DCL	DCU
B	01 (1)	1001 (9)
B1	1110 (14)	
B2	110 (6)	
B4	100110(38)	
B5	100 (4)	

Table 2. SADL and SADU.

Blocks	SADL	SADU
B,B1	13	23
B1,B2	5	3
B1,B4	37	47
B1,B5	3	5

B1= {12, 10, 12, 12},
B2= {2, 2, 4, 2},
B3 = {5, 6, 4, 2},
B4 = {30, 33, 36, 63},
B5= {4, 5, 6, 3},
The calculation of the SAD between the reference block and each block in the search window is as follows:

SAD(B,B1) = 11+9+8+10=38,
SAD(B,B2) = 1+1+1+0= 3,
SAD(B,B3) = 4+5+1+0=10,
SAD(B,B4) = 29+32+33+61=155,
SAD(B,B5) = 3+4+3+1= 11.

As shown, block B2 is the best match for B. Using our algorithm, let's assume that our start point is B3. Thus, the first step is to calculate the actual SAD(B, B3) = 4+5+1+0=10.
Blocks in binary format:

B = {1, 1, 3, 2}
= {01, 01, 11, 10}
B1= {12, 10, 12, 12}
= { 1100, 1010, 1100, 1100}
B2= {2, 2, 4, 2}
= {010, 010, 100, 010}
B3 = {5, 6, 4, 2}
= {101, 110, 111, 010}
B4 = {30, 33, 36, 63}
= {011110, 100001, 100110, 111111}
B5= {4, 5, 6, 3}
= {100, 101, 110, 011}

Calculate DCU and DCL for all blocks, as shown in Table 1.
Calculate SADL and SADU for all of the other blocks with respect to block B, as shown in Table 2.

B1 and B4 will be excluded because SADL(B, B1) and SADL(B, B4) are 13 and 37, respectively, which are greater than SAD(B, B3). Since these values are the lower limits for SAD(B, B1) and SAD(B, B4), this implies that SAD(B,B1) and SAD(B, B4) are both greater than SAD(B, B3). The new search window will include only B2 and B5, and the method will have to calculate their actual SAD values to determine the best match.
This example shows that half the search window can be excluded from the further computation of the actual SAD, which is a very expensive operation.

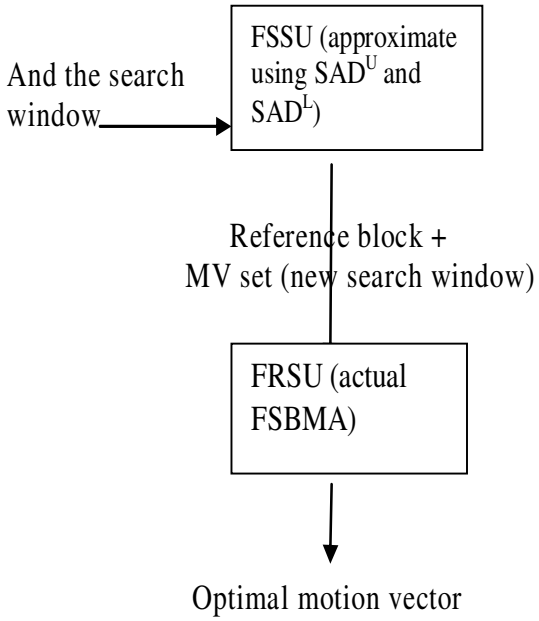

Figure 2. The proposed architecture.

Low-power vlsi architecture

In this section, we present a VLSI architecture for the two-stage motion estimation algorithm. Power consumption savings are achieved because SADU and SADL are computationally inexpensive compared to the actual SAD computation. When the number of candidate blocks eliminated in the interval-based matching stages is high, fewer actual SAD computations will have to be carried out. Consequently, there is less power consumption. Also, due to the reduced number of gates required to perform addition without carrying propagation, there is less required hardware compared to that for the actual SAD computation.

The VLSI architecture consists of two main units, namely (a) the First Step Search Unit (FSSU) and (b) the Full Resolution Search Unit (FRSU). The first unit is an interval-based matching stage that uses approximate calculations for the SAD. The second unit is the full-resolution unit that calculates the actual SAD for the candidate blocks that have not been eliminated. The final motion vector is generated in this stage. Figure 2 shows the proposed architecture.

First step search units

The FSS unit is an interval-based matching stage and uses the

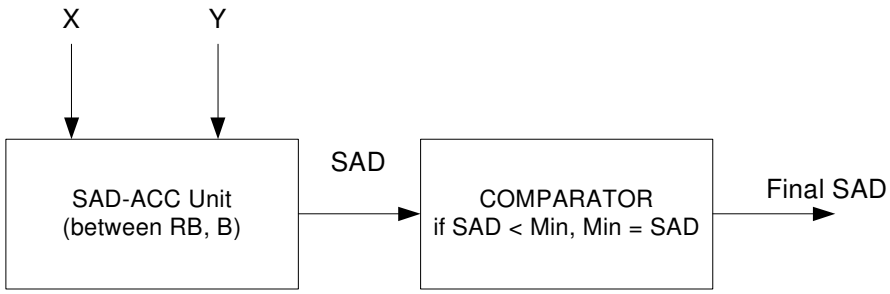

Figure 3. The block diagram of the FRSU.

SADU and SADL approximations. The reference block and candidate block are supplied to the SADU and SADL modules. These modules compute the approximate functions SADU and SADL for all of the candidate blocks in the search window. As a new exact SAD is calculated between the reference block and one of the blocks in the search window, the minimum is stored in the SMIN register. All of the approximate SADs are calculated and stored in a temporary buffer. The length of the buffer is equal to the number of candidate blocks in the search window.

The elimination of the candidate blocks is done by comparing all of the approximate SADs stored in the temporary buffer to the SMIN in the comparator. The output of the comparator is a one bit output indicating whether the candidate block has been eliminated or kept. This is stored in the MV set register, whose length is the same as the search window.

The full resolution search unit: FRSU

The FRSU calculates the actual SADs between the reference block and the candidate blocks in the MV set. The main circuit in FRSU is the SAD-accumulate unit, which computes the SAD function. The minimum SAD is found and the optimum motion vectors are obtained in this unit. The block diagram of the FRSU is depicted in Figure 3.

EXPERIMENTAL RESULTS

The performance of the proposed motion estimation algorithm PFSBM was evaluated against several motion estimation algorithms. The list of the algorithmsinvolved in the performance evaluation includes the hierarchical diamond search (HDS) proposed in Urban et al. (2009), the low power, FPSoC-based architecture for a fast ME algorithm in H.264/AVC (Ndili and Ogunfunmi, 2009), the adaptive content-based subsample algorithm (ACSA) (Cheng and Dung, 2005), the successive elimination algorithm (SEA) (Li and Salar, 1995), the multilevel successive elimination strategy (MSEA) (Gao et al., 2000), and the extended SEA (ESEA) (Brnig and Niehsen, 2001). All of the algorithms included in the comparison were implemented in the raster scan order as originally implemented (Liu et al., 2008). Our proposed algorithm was implemented in the raster scan order and in spiral order. We experimented using a spiral order scan starting from the original place of the candidate block, which was found to enhance the performance by eliminating more candidate blocks. Our comparison focused on the computational costs. Tables 3 and 4 tabulate the experimental results for the total number of eliminated blocks when applying our proposed PFSBM and the other motion estimation algorithms to six different video sequences.

The computation cost is one of the most important parameters for a lip reading authentication system. It determines both the speed and the anticipated power consumption of VLSI implementations of proposed algorithms. To evaluate the performance, the computational cost was defined as the number of candidate blocks eliminated using the proposed algorithm. From the experimental results, we can see that the proposed PFSBM algorithm consistently outperformed the other motion estimation techniques for experiments on different video sequences. Table 3 shows the average number of eliminated candidate search points as a percentage of the number of blocks in the search window over 40 frames. On average, the proposed algorithm eliminated more than 80% of the candidate blocks in the search window, while the conservative approximation algorithm eliminated 22% of the candidate blocks. The speedup advantage of the proposed algorithm over both the conservative approximation and FSBMA are also presented. The results of the experiments indicate that the proposed algorithm obtained a speed improvement over the other algorithms by eliminating many more blocks from the search window and by calculating SADU and SADL from DCU and DUL, which are calculated once per block and used several times. In addition, a performance enhancement was observed when the

Table 3. Average number of candidate blocks per reference block eliminated for several lip reading single digit video sequences for a 16 X 16 search window.

Digit	1	2	3	4	5	6
PFSBM spiral scan	223.9	214.6	218.33	234.75	236.08	231.2
PFSBM raster scan	177.9	189.1	172.22	178.3	183.0	179.23
MSEA (Li and Salari, 1995)	140.22	145.4	148.27	146.46	142.3	139.5
ESEA (Gao et al., 2000)	168.3	162.52	166.21	158.3	167.23	152.12
SEA (Brnig and Niehsen, 2001)	43.4	40.58	46.63	44.08	56.69	42.36
HDS(Urban et al., 2009)	167.1	179.17	162.42	182.32	173.0	189.2
FPSoC (Ndili and Ogunfunmi, 2009)	172.32	163.11	181.29	156.6	162.43	169.52
ACSA (Cheng and Dung, 2005)	158.3	167.23	166.21	176.8	152.78	152.12

Table 4. Average number of candidate blocks per reference block eliminated for several lip reading video sequences of three digits for a 16 x 16 search window.

Digits	"123"	"124"	"134"	"431"	"527"	"618"
PFSBM Spiral scan	190.45	193.0	195.45	200.27	202.78	188.72
PFSBM raster scan	157.19	159.11	162.42	158.31	143.01	169.23
MSEA (Li and Salar, 1995)	120.22	125.41	118.7	116.62	122.13	131.15
ESEA (Gao et al., 2000)	138.31	132.2	141.21	128.27	137.28	132.02
SEA (Brnig and Niehsen 2001)	31.14	29.08	36.23	34.48	26.9	32.3
HDS (Urban et al., 2009)	137.19	139.14	122.92	132.69	133.06	119.28
FPSoC (Ndili and Ogunfunmi, 2009)	122.87	113.45	131.39	116.68	122.72	114.52
ACSA (Cheng and Dung, 2005)	128.39	127.78	136.77	136.26	132.43	122.33

spiral scan was used for the proposed algorithm.

Table 4 shows the average number of search points eliminated over 260 frames using consecutive digits. From the experimental results, we can see that the proposed algorithm consistently outperformed the other optimal motion estimation techniques for experiments on different video sequences.

Tables 3 and 4 also compare the proposed algorithm to the non-optimal algorithms presented in (Urban et al. 2009; Ndili and Ogunfunmi, 2009; Cheng and Dung, 2005). Our technique also outperformed them, but not to the extent it outperformed the optimal algorithms.

Conclusions

This paper proposed a fast and optimal motion estimation algorithm, PFSBM. We demonstrated the superior performance of the proposed algorithm compared to previous optimal motion estimation methods through experiments on video sequences. Furthermore, we enhanced the performance of the proposed algorithm by using spiral scan instead of raster scan. The presented algorithm is a pruning algorithm for full-search block-matching that reduces power consumption without any loss in the accuracy of the results.

REFERENCES

Ahn TJ, Moon YH, Kim JH (2004). Fast full-search motion estimation based on multilevel successive elimination algorithm. IEEE Trans. Circuits Syst. Video Technol., 14(11): 1265-1269.

Brnig M, Niehsen W (2001). Fast full-search block matching. IEEE Trans. Circuits Syst. Video Technol., 11(2): 241–247.

Cheng HW, Dung LR (2005). A content-based methodology for power-aware motion estimation architecture. IEEE Trans. Circuits Syst., 52(10): 631-635.

Cheung CH, Po LM (2005). Novel cross-diamond-hexagonal search algorithms for fast block motion estimation. IEEE Trans. Multimed., 7(1): 16–22.

Gao XQ, Duanmu CJ, Zou CR (2000). A multilevel successive elimination algorithm for block matching motion estimation. IEEE Trans. Image Process., 9(3): 501–504.

Li W, Salari E (1995). Successive elimination algorithm for motion estimation. IEEE Trans. Image Process., 4(1): 105–107.

Liu SW, Wei SD, Lai SH (2008). Fast Optimal Motion Estimation Based on Gradient-Based Adaptive Multilevel Successive

Elimination. IEEE Trans. Circuits Syst. Video Technol., 18(2): 263–267.
Mahmoud H, Alghathbar KS, Muhaya FB (2009). Motion estimation analysis for unsupervised training for lip reading user authentication systems. Int. Conf. Autom. Inf, Prague, Czech Republic, pp. 80–87.
Ndili O, Ogunfunmi T (2009). FPSoC Based Architecture for A Fast Motion Estimation Algorithm in H.264/AVC. Technical Report, Department of Electrical Engineering, Santa Clara University.
Urban F, Nezan JF, Raulet M (2009). HDS, a real-time multi-DSP motion estimator for MPEG-4 H.264 AVC high definition video encoding. J. Real-Time Image Proc., 4(1): 23–31.