# EFBLA: a two-phase matching algorithm for fast motion estimation

H.-W. Cheng and L.-R. Dung

A novel motion estimation algorithm that contains edge matching and block matching is presented. The edge matching reduces candidates for the block matching that performs accurate searching. Thus, the two-phase procedure speeds up motion estimation. As a result, edge-matching first block-matching last algorithm (EFBLA) may significantly save the computation load by 93.9%.

*Introduction:* The most popular algorithm for the VLSI implementation of motion estimation is the block-based full-search algorithm [1–3]. However, the full-search algorithm notoriously needs high computation load and large memory size. The high computational cost has become a major problem in the implementation of motion estimation. Many papers have proposed different ways to reduce the computation requirement of the full-search algorithm. Most of them target on the elimination of impossible motion vectors, such as SEA [4] and LRQ [5], and only perform motion estimation on the possible ones. Applying this philosophy, we have developed a two-phase algorithm that is suitable for implementation of motion estimation. Our goal is to decrease the number of block-matching evaluations without degrading the video quality such that the computation load can be significantly reduced. The algorithm, called the edge-matching first block-matching last algorithm (EFBLA), first employs the edge-matching procedure to determine candidate motion vectors and then performs the conventional block matching with the SAD criteria on the candidates. The edge-matching procedure does not require complex computation; it only needs shift operations, quantisation, comparison and thresholding. Therefore, our algorithm requires fewer operations and lower memory size compared with the full-search algorithm. As a result of benchmarking and comparing to the full-search algorithm, EFBLA may significantly save the computation load by 93.9% while the degradation of PSNR is very little.

*Algorithm:* The edge-matching procedure first performs highpass filtering on a macro-block of the current frame, called a reference block, to determine edge pixels. Then, we start matching the reference block with macro-blocks of the previous frame, called target blocks. The matching criterion is the unmatched edge pixel count (UEPC). An unmatched edge pixel is a pixel of the target block whose low-resolution quantised value is different from that of the corresponding edge pixel of the reference block. The EFBLA only picks the motion vectors with lower UEPC target blocks as the survived motion vectors (SMVs). Following the edge-matching phase, the proposed algorithm begins to perform block matching with the SAD criteria on SMVs.

The first phase of EFBLA contains five steps as described below. Assume that the macro-block size is $N$-by-$N$ and the searching window is $2p$-by-$2p$; both co-ordinates are in the range $-p$ to $p-1$. The orientation of the reference block is $(x, y)$.

*Step 1: Perform basic highpass spatial filtering on the reference block:* Step 1 first performs the edge enhancement using a highpass spatial filter mask, as shown in (1). In (1), the $f_k(x_i, y_i)$ represents the intensity of the pixel at $(x_i, y_i)$ in the reference block. The larger the value of $|\nabla f_k(u, v)|$ the more possible is the pixel on the edge:

$$\nabla f_k(u,v) = \sum_{i=-1}^{1}\sum_{j=-1}^{1} c(i,\ j) f_k(u+i, v+j)$$

where

$$\begin{cases} c = 8, & \text{when } (i,\ j) = (0,0) \\ c = -1, & \text{otherwise} \end{cases} \tag{1}$$

$$(u,v) = (x,y) \simeq (x+N-1, y+N-1)$$

*Step 2: Calculate the edge threshold and mark the edge pixels:* The EFBLA uses (2) to calculate the edge threshold ($Eth$) and marks the edge pixels as shown in (3).

$$Eth = \frac{\max(|\nabla f_k(u,v)|) + \min(|\nabla f_k(u,v)|)}{2} \tag{2}$$

$$\alpha(u,v) = \begin{cases} 1 & \text{if } |\nabla f_k(u\pm 1, v\pm 1)| > Eth \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

*Step 3: Determine the scan direction:* The data reusability is highly dependent on the scan direction. For instance, if the edge pixels are widely distributed along the $y$-co-ordinate, searching along the $y$-co-ordinate can reuse the data efficiently. The EFBLA has two scan directions: column-by-column and row-by-row. To make the decision of scan direction, this step first determines the span width of edge pixels with $x$-co-ordinate, named the $x$-span, and the span width of edge pixels with $y$-co-ordinate, named the $y$-span. If the $x$-span is smaller than the $y$-span, the step selects the column-by-column scan as the direction; otherwise, the scan direction will be row-by-row.

*Step 4: Quantise the edge pixels of the reference block:* This step quantises the pixel values at the edge pixels for the low-resolution computation. Equation (4) represents the quantisation of the reference blocks where $\hat{f}_k(u, v)$ is the value of two most significant bits (MSBs) of $(f_k(u, v) - Avg_k)$ where $Avg_k$ is the total pixel average of the reference block:

$$\hat{f}_k(u,v) = Q_2(f_k(u,v) - Avg_k)\forall\alpha(u,v) = 1$$

where

$$Avg_k = \frac{\sum_{u=x}^{x+N-1}\sum_{v=y}^{y+N-1} f_k(u,v)}{N^2} \tag{4}$$

*Step 5: Perform edge matching and generate SMVs:* The step starts performing edge matching in the searching area along with the scan direction obtained by step 3. The edge matching employs the criteria of UEPC, as shown in (5). In (5), $\hat{f}_{k-1}(u+dx, v+dy)$ is the quantisation result of the target block with the motion vector $(dx, dy)$:

$$UEPC(dx,dy) = \sum_{u=x}^{x+N-1}\sum_{v=y}^{y+N-1} \alpha(u,v)\cdot\delta[\hat{f}_k(u,v), \hat{f}_{k-1}(u+dx, v+dy)] \tag{5}$$

where

$$\hat{f}_{k-1}(u+dx, v+dy) = Q_2(f_{k-1}(u+dx, v+dy) - Avg_k)\forall\alpha(u,v) = 1$$

and

$$\delta\left[\hat{f}_k, \hat{f}_{k-1}\right] = \begin{cases} 0, & \text{for } \hat{f}_k = \hat{f}_{k-1} \\ 1, & \text{otherwise} \end{cases}$$

Next, the step picks the motion vectors with the lowest two UEPC target blocks as the survived motion vectors (SMVs) for each scan line, either row or column.

Following the first phase, the EFBLA performs block-matching with the SAD criteria on SMVs. Note that the block-matching requires much fewer evaluations than the traditional full-search block-matching because the first phase has eliminated a large amount of impossible motion vectors.

*Results:* The proposed algorithm significantly reduces the number of motion vectors that require costly evaluations. To compare with the other motion estimation algorithms, this Letter uses two metrics: computation cost and the peak signal-to-noise ratio (PSNR). Since the major operation of motion estimation algorithms is the addition, we consider approximately the total number of additions required for each macro-block as the computation cost.

We have used four MPEG video clips, 'fish', 'weather', 'news' and 'children', as the testbenches. Each frame has $352 \times 288$ pixels and each pixel value is 8-bit. The macro-block size is $16 \times 16$, and the search window ranges from $(-16, -16)$ to $(15, 15)$.

Tables 1 and 2 shows the simulation results for three algorithms: full-search algorithm (FS), low-resolution quantisation (LRQ) and the proposed algorithm. We simulated 40 frames for each testbench. Obviously, the EFBLA saves up to 93.9% of the computation cost of

FS while the PSNR degradation is less than 0.24 dB. Fig. 1 demonstrates that the quality of the EFBLA is very close to the quality of the others. The results illustrate that the proposed algorithm is capable of speeding up the motion estimation as well as having good quality. Compared with LRQ, the proposed algorithm also outperforms in terms of computation cost. As shown in Table 1, the EFBLA has 7–15% reduction of computation cost. Regarding the PSNR, both algorithms are compatible with each other.

**Table 1:** Comparison of number of operations

| | Fish | Weather | News | Children |
|---|---|---|---|---|
| FS | 783 360 | | | |
| LRQ | 57 279 | | | |
| Proposed | 53 263 | 50 456 | 47 541 | 48 797 |
| Reduction compared to FS | 93.2% | 93.6% | 93.9% | 93.8% |
| Reduction compared to LRQ | 7.0% | 11.9% | 17.0% | 14.8% |

**Table 2:** Comparison of average PSNR (dB)

| | Fish | Weather | News | Children |
|---|---|---|---|---|
| FS | 29.5758 | 32.7005 | 36.3478 | 26.3535 |
| LRQ | 29.4117 | 32.6307 | 36.0524 | 26.1839 |
| Proposed | 29.3986 | 32.4642 | 36.2031 | 26.1330 |

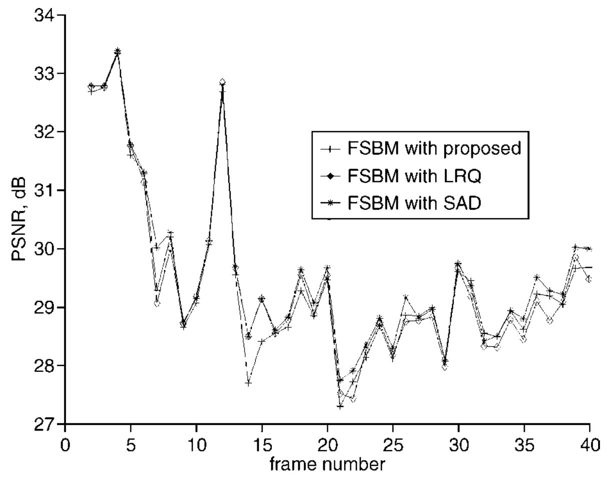

**Fig. 1** *PSNR of 'fish' sequence*

*Conclusion:* A novel algorithm to significantly speed up the motion estimation by reducing the evaluation of motion vectors is proposed. As a result of simulating video clips, EFBLA significantly saves the computation load by 93.9% while the quality degradation is very little compared with FS, less than 0.24 dB.

© IEE 2004 *3 February 2004*
*Electronics Letters* online no: 20040476
doi: 10.1049/el:20040476

H.-W. Cheng and L.-R. Dung (*Department of Electrical and Control Engineering, National Chiao Tung University, Hsinchu, Taiwan, Republic of China*)

## References

1 Shen, J.-F., Wand, T.-C., and Chen, L.-G.: 'A novel low-power full-search block-matching motion-estimation design for H.263+', *IEEE Trans. Circuits Syst. Video Technol.*, 2001, **11**, (7), pp. 890–897
2 Do, V.L., and Yun, K.Y.: 'A low-power VLSI architecture for full-search block-matching motion estimation', *IEEE Trans. Circuits Syst. Video Technol.*, 1998, **8**, (4), pp. 393–398
3 Brunig, M., and Niehsen, W.: 'Fast full-search block matching', *IEEE Trans. Circuits Syst. Video Technol.*, 2001, **11**, (2), pp. 241–247
4 Li, W., and Salari, E.: 'Sucessive elimination algorithm for motion estimation', *IEEE Trans. Image Process.*, 1995, **4**, (1), pp. 105–107
5 Lee, S., and Chae, S.I.: 'Motion estimation algorithm using low-resolution quantization', *Electron. Lett.*, 1996, **32**, (7), pp. 647–648