DSRA: A BLOCK MATCHING ALGORITHM FOR NEAR-REAL-TIME VIDEO ENCODING

Chun-Hung Lin[1], Ja-Ling Wu[1,2] and Yi-Shin Tung[1]
[1]Communication and Multimedia Lab.
Department of Computer Science and Information Engineering
National Taiwan University, Taipei, 106, Taiwan, R.O.C.
[2]Department of Information Engineering
National Chi Nan University
Puli, 545, Taiwan, R.O.C.

ABSTRACT —

The correlations of motion vectors between neighboring image blocks have long been applied to improve the efficiency of block matching algorithms. In this paper, a block matching algorithm with a dynamic search range is presented. The search range is adjusted based on the correlations of the motion vectors between neighboring blocks. It is shown by simulation results that the proposed method can speed an MPEG-1 encoding system up to a ratio of 13.3 and increases the bit-rate only of 0.025 bpp, in average. It is therefore beneficial to embed the proposed method into a motion-based encoding system.

1 INTRODUCTION

Because of the promotion of the computing power, general-purpose CPUs have been largely applied for realizing real-time software-based video players (decoders). So far, there are many commercial products of software-based MPEG-1 [1] players available in the market. They are used to display MPEG-1 encoded video sequences in real-time without any additional hardware support. The major advantages of using pure software video codecs are: (1) the cost is quite low while comparing with the hardware solutions, (2) the portability of software makes them easier to be transported to different platforms. Due to the rapid progress of the performance of general-purpose CPUs, it is a current trend to implement the video codec by using software only.

The most important three functional blocks of a typical video encoding system are motion estimation (ME), discrete cosine transform (DCT), and variable length coder (VLC). Lots of methods have been proposed [2]–[5] for implementing the DCT. The computational complexity of the DCT is therefore not a major concern of the whole encoding process. As for the VLC, because it is basically realized by using look-up-table operations, the space for improving the coding performance is limited. When the computational complexity of the other two components has been well controlled, the most important issue for real-time software-based video encoding becomes to enhance the efficiency of the ME component. For example, in DEC's implementation of an MPEG software-based encoder, the computational complexity of ME is more than 40% of that of the overall encoding system [6]. The reduction of the computational complexity of ME is, therefore, the main subject of this paper.

While developing a pure-software video encoding system, it is impossible to apply the full search algorithm (FSA) for implementing the ME component because of its high computational complexity. Many fast ME algorithms have been proposed in the literature. Most of the traditional block matching algorithms (BMAs) [7]–[12], by means of heuristic searching, are based on the assumption that matching errors increase mononically from the minimum within a fixed search range. Among these algorithms, three-step search (TSS) and two-dimensional logarithmic search (TDL) are often used. However, the performance of these algorithms is not as good as expected because the above assumption is not always true. There still exists space for finding more efficient BMAs.

The high correlation between neighboring blocks in an image has been investigated to improve the perfor-

Contributed Paper
Manuscript received January 31, 1997

mance of various kinds of coding algorithms. It is also observed that the motion vectors of neighboring blocks are very similar because each object in the video scene usually covers many image blocks. Some previously proposed algorithms, based on inter-block motion vector correlations, were proved to be very efficient [13]–[15]. In these algorithms, several candidate motion vectors, which are selected from the estimated motion vectors of the neighboring blocks, are checked and the one with the minimum matching error is chosen to be a feasible solution. If the matching error of the feasible motion vector is larger than a predefined threshold, a refinement stage has to be employed. In Hsieh's work [13], a full search centered at the feasible motion vector is performed. Unfortunately, in this approach, it is difficult to determine an appropriate search range. If the search range is large, a long execution time is unavoidable (which makes the BMA inefficient); however, if the search range is too small, the estimation result will be poor. In [15], the image blocks are classified into three different groups. Different selection schemes of neighboring blocks are applied in each group of blocks for determining the candidate motion vectors. In the refinement stage, a local search scheme (which is similar to the work proposed in [16]) is performed around the feasible motion area. Although the performance of the algorithms is shown to be impressive, the system complexity of these algorithms is much higher than that of the other BMAs. Therefore, it is difficult to implement these algorithms in hardware, or even in software.

In this paper, based on a similar strategy, an efficient BMA (called the dynamic search range algorithm (DSRA)), is proposed. In the DSRA, a full search is performed around the feasible motion area but the search range can be dynamically adjusted so that the estimation performance will not degrade much while the computational complexity is reduced largely.

2 DYNAMIC SEARCH RANGE ALGORITHM

In the DSRA, a full search centered at the initial search point is performed. The search range is dynamically adjusted in block-by-block sense so that the true motion can be found within a limited number of block matching.

Let $\hat{m} = (\hat{v}_x, \hat{v}_y)$ represent the initial motion vector, and $r = (b_x, b_y)$ be the search range of the current estimated block B. The best motion vector can be resolved by performing a full search within the search area specified by $\hat{m}$ and r. The best motion vector $\bar{m}$ is defined as

$$\begin{aligned} \bar{m} &= \arg_{(\hat{v}_x+d_x, \hat{v}_y+d_y)} \min SAD(\hat{v}_x + d_x, \hat{v}_y + d_y), \\ & \quad -b_x \le d_x \le b_x, -b_y \le d_y \le b_y, \end{aligned} \tag{1}$$

and

$$SAD(v_x, v_y) = \sum_{(i,j)\in B} |I_n(i,j) - I_{n-1}(i + v_x, j + v_y)|, \tag{2}$$

where $I_n(i,j)$ denotes the intensity of the pixel at location (i,j) in the nth frame.

In the previously proposed algorithms which have taken motion correlation into account, the candidate motion vectors were usually selected from the motion vectors of the neighboring blocks. Each of the candidate motion vectors is examined, and the one with the minimum matching error is chosen as the initial motion vector $\hat{m}$. The number of candidate motion vectors dominates the required block matching number for deciding $\hat{m}$. It is expected that $\hat{m}$ will approach the true motion vector and the required block matching number in the refinement stage would be small, so that the total computational load is not high.

In the DSRA, $\hat{m}$ is directly assigned to be the motion vector of the most recently estimated neighboring block. Because a full search with an appropriate search range will be performed later in the refinement stage, it is unnecessary to examine a lot of candidate motion vectors. While for slow- to medium-motion video sequences, the average block matching number for each block could be considerably reduced. Moreover, only a few previously estimated motion vectors have to be stored in the buffer as compared to the traditional correlation-based approaches.

The initial motion vector of the (i,j)th block, in the DSRA, is selected as

$$\hat{m}_{i,j} = \bar{m}_{\rho(i,j)}, \tag{3}$$

where ρ represents the index of the most recently estimated neighboring block, i.e.,

$$\rho(t) = \begin{cases} (i-1,j), & i > 0, \\ (i,j-1), & i = 0, \end{cases} \tag{4}$$

where $t = (i,j)$.

In order to capture the true motion within an economical block matching number, the search range has to be adjusted according to the violence level of motion. In a slow moving or motion-homogeneity region, the search range is shrunk to reduce the computational complexity. Whereas, in a violently moving or motion-extraneousness region, where $\hat{m}$ might be far from the true motion, the search range is stretched to catch the true motion.

The search range, in the DSRA, is adjusted as

$$b_{x_{i,j}} = b_{x_{\rho(i,j)}} + D_x(i,j), \tag{5}$$

$$b_{y_{i,j}} = b_{y_{\rho(i,j)}} + D_y(i,j), \tag{6}$$

where D_x and D_y represent range difference values of the x- and y- axes, respectively. In a violently moving region, the range difference values should be higher than that of a slow moving region. In our implementation, the range difference values are set as

$$D_x(i,j) = \Delta(\bar{v}_{x_{\rho(i,j)}}, \bar{v}_{x_{\rho(\rho(i,j))}}, b_{x_{\rho(i,j)}}), \tag{7}$$

$$D_y(i,j) = \Delta(\bar{v}_{y_{\rho(i,j)}}, \bar{v}_{y_{\rho(\rho(i,j))}}, b_{y_{\rho(i,j)}}), \tag{8}$$

and

$$\Delta(\alpha,\beta,\gamma) = \begin{cases} 1, & |\alpha-\beta| = \gamma, \\ 0, & |\alpha-\beta| = \gamma+1, \\ -1, & |\alpha-\beta| > \gamma+1, \end{cases} \tag{9}$$

where $(\bar{v}_{x_{i,j}}, \bar{v}_{y_{i,j}})$ denotes the motion vector of the (i,j)th block.

While computing the matching error between the current block and the corresponding block of the testified motion vector, the absolute difference of every two matched pixels contributes to the matching error. Each time a difference value is added to the computed matching error, the currently accumulated value is checked. If the currently accumulated matching error is larger than the minimum matching error obtained so far, the computation of the matching error against the testified block is stopped and the computation of the matching error for the next block is proceeded. This 'early jump out' technique will speed up the searching process a lot. The pseudo codes of this technique can be listed as follows:

```
current_error = 0
for each pixel (i,j) ∈ testified block B
    begin
        current_error = current_error +
            |I_n(i,j) − I_{n−1}(i + υ_x, j + υ_y)|
        if current_error ≥ min_error
            go to EJO
    end
min_error = current_error
best motion vector m̄ = (υ_x, υ_y)
EJO:
proceed to the next motion vector
```

Whereas, the typical block matching is proceeded as follows:

```
current_error = 0
for each pixel (i,j) ∈ B
    current_error = current_error +
        |I_n(i,j) − I_{n−1}(i + υ_x, j + υ_y)|
if current_error < min_error
    begin
        min_error = current_error
        best motion vector m̄ = (υ_x, υ_y)
    end
proceed to the next motion vector
```

When the full search is completed, the so-obtained minimum matching error is compared to a predefined threshold. If the minimum matching error is smaller, the corresponding motion vector is selected as the solution. Otherwise, another full search with a predefined maximum search range is performed to find the true motion vector, where an alternating pixel subsampling

scheme [17] is applied to reduce the computational complexity.

The whole algorithm is depicted in Fig. 1 and is summarized as follows:

1) Initially, the search range is set to be $(b_x, b_y) = (1, 1)$ and the previous motion is set to be zero, that is $(\hat{v}_x, \hat{v}_y) = (0, 0)$.

2) The matching errors of the zero motion vector $(0, 0)$ and the other candidate motion vectors $(\hat{v}_x + d_x, \hat{v}_y + d_y), where - b_x \leq d_x \leq b_x$ and $-b_y \leq d_y \leq b_y$, are calculated. Let e_1 represents the minimum matching error among them. If e_1 is larger than the threshold T, go to Step 4.

3) Adjust the search range of the next block according to (5) and (6), and go to Step 6.

4) In order to find the true motion, a full search with an alternating $4:1$ pixel subsampling [17] is performed, where the search range is set to be a predefined value.

5) The search range of the next block is set to be $(b_x, b_y) = (1, 1)$.

6) Proceed to the next block and go to Step 2.

3 EXPERIMENTAL RESULTS AND DISCUSSIONS

To compare the performance of the proposed algorithm with the other ones, six standard video sequences are tested. The basic information of the sequences is listed in Table 1. In the simulations, four different BMAs: the FSA, the TSS, the spatial-2 algorithm (S2, which was briefly described in Section 1) [15], and the proposed DSRA are implemented for comparison. The estimated block size is 16×16 pixels, and the predefined maximum search range of the DSRA and the other BMAs are -16 to 16 pixels.

In Table 2, the average block matching numbers required for each algorithm are shown. For most of the sequences, the block matching numbers of the DSRA are always less than that of the other methods. It can be seen from Table 3 that a large amount of block matching numbers have been saved by applying the DSRA. While comparing with the FSA, the DSRA needs, in average, only 1.75% block matching of the previous one. In the TSS, the average block matching numbers are stable and ranged from 30 to 31. This is because the searching pattern is static and basically independent of the content of the video sequences. Whereas, in the other two fast algorithms, which have taken the correlations of neighboring motion vectors into account, the block matching numbers depend largely on the content of the test video sequences. For most of the test video sequences, fewer block matching numbers are required due to the assistance of the correlation of the neighboring motion vectors. The only exception is the "Football" sequence. In that sequence, there are many objects moving violently; therefore, the correlation of motion vectors between neighboring blocks is not high. In this case, the computations will not be reduced. Furthermore, it can be seen from Table 4 that the average PSNR values of the predicted "Football" frames obtained by using the other two fast BMAs are worse than that of the TSS. Nevertheless, because the difference values of the corresponding pixels in the two matched blocks are not all calculated both in the S2 and the DSRA, the execution time of these two methods is always less than that of the TSS, as shown in Table 5.

It is known from Tables 2 and 4 that the DSRA needs less block matching, and the average PSNR values of the predicted pictures using the DSRA is very close to or even better than those of the other two fast algorithms. Especially, for the sequences with monotonous motion, where the correlation of motion vectors between neighboring blocks is very high, the benefit of the DSRA becomes very prominent. For example, in the "Mobile" sequence, the average block matching number is only 10.47 while the average PSNR is only 0.12 dB less than that of the FSA.

To compare the computational complexities of different BMAs, in addition to block matching numbers, the control overheads have to be considered. However, it is not easy to compare the control overheads of different

algorithms because they depend heavily on the practical implementation. In our simulations, the BMAs are implemented on a SUN Ultra SPARC-167 workstation. The speed-ups of different fast algorithms against the FSA are shown in Table 5. It can be seen that the speed-ups of the DSRA are higher than the other algorithms for most of the test sequences.

Because a video encoder consists of some other components besides ME, the performance of different BMAs might be different when they are applied to a whole video encoding system. In a video encoding system, the referenced frames of ME are usually generated from the decoder. The decoded/reconstructed frames are not the same as the original ones due to quality degradation caused by compression, so the correlations of neighboring motion vectors are different. Moreover, the estimated results of ME will affect the computational complexity of the other components in a video encoding system, such as block-type classification and variable length coding. It might be possible that the overall computational complexity of the encoding system increases because of the increasing computational complexity for the other components, which is caused by the reduction of the computations of ME.

In order to see the practical efficiency of different BMAs, they are embedded into two MPEG-1 encoders: encoder-I[1] constructed on a SUN Ultra SPARC-167 workstation and encoder-II[2] constructed on a Pentium 180 personal computer. The encoding parameters are adjusted to ensure that the picture qualities of the decoded video in the individual encoder by applying different BMAs are nearly the same. The experimental results of the encoding frame-rate and bit-rate are shown in Tables 6–9.

Because the MPEG encoding method is not specified in the MPEG video standard [1], different MPEG encoders (with different coding strategies) can be implemented to produce valid MPEG bitstreams. The simulations are therefore performed on two different encoders.

On the other hand, the efficiency of the encoding programs depends largely on the environments of the operating systems and the program compilers. Hence, two different platforms with different compilers (GNU C compiler versions 2.7 and Microsoft VC++ compiler versions 4.2) are used in our simulations. In encoder-II, the BMAs with candidate matched blocks of succeeded indices are easier to be optimized by the compiler. It can be seen from Tables 6 and 7 that the FSA and the DSRA gain more from the optimizer.

There are many differences between the coding strategies of the two adopted encoders. For example, an "open-loop" structure is adopted in encoder-I, that is, the referenced frames of ME are directly taken from the original video sequences, while the "close-loop" structure is applied in encoder-II where the decoded frames are taken as the references of ME. When the decoded frames are referenced, the motion correlations of neighboring motion vectors degrade, so the estimation accuracy of the BMAs by means of such correlations decreases. As shown in Table 9, the bit-rates of applying S2 and the DSRA are more than that of the TSS. Whereas, in encoder-I (as shown in Table 8), more bit-rates are required for the TSS. Nevertheless, the prediction error is accumulated when the open-loop structure is used, which brings much quality degradation to the rear frames of each group-of-picture unit.

In addition to the coding structures, the classification strategies of block types are also different in the two encoders. In encoder-I, a more delicate strategy is used so that more blocks can be classified as P- or B-type blocks. Fewer bit-rates are hence required in encoder-I. Especially, for the head-and-shoulder sequences (e.g., "Salesman" and "Miss America"), where almost half of a picture belongs to the background area and therefore, most blocks are classified to be the skipped blocks, so that many computations are saved. As for the other sequences, the computations needed to classify the block types become an overhead. It can be seen from Table 6

[1]MPEG-1 video software encoder, developed by L. A. Rowe, K. Gong, K. Patel, and D. Wallach, Computer Science Division-EECS, Univ. of Calif. at Berkeley.

[2]MPEG-1 video software encoder, developed by Communication and Multimedia Lab., Department of Computer Science and Information Engineering, National Taiwan University.

that the frame-rates of the head-and-shoulder sequences are much higher than the other sequences.

From Tables 6 and 7, it can be seen that the coders embedded with the DSRA are always faster than the other ones. As shown in Table 6(b), the encoding frame-rate of encoder-I (on the SUN workstation) by applying the DSRA can reach nearly 30 frames per second for the head-and-shoulder video sequences. In Table 7(b), the encoding frame-rate of encoder-II (on PC) can reach above 17 frames per second for different video sequences. Because the computing power of CPUs will be improved continuously (for example, Pentium Pro 200 and MMX have recently been released), it is expected that a real-time software-based MPEG encoder (by applying the DSRA) will be constructed in the near future.

While comparing with the TSS, the frame-rate of applying the DSRA is better. In encoder-I, the speed-ups of applying the DSRA in CIF and QCIF resolutions are 0.94 and 4.40 higher than their TSS counterparts, and the encoded bit-rates are 0.0306 bpp and 0.0008 bpp less. In encoder-II, the speed-ups of applying the DSRA are 2.49 and 9.54 higher, while the bit-rates are 0.2072 bpp and 0.1525 bpp more (this comes from the different implementation strategies adopted in encoder-II, as mentioned above).

The bit-rates generated by the S2-based coder are lower than that of the DSRA-based one but the former requires more execution time. In average, the bit-rates of the coder using S2 are 0.0318 bpp less than that of the DSRA, while the speed-up of the later is 3.35 higher than the former. Because the control complexity of the DSRA is much lower than that of the S2, the DSRA gains much from the program compiler in encoder-II. This explains why the encoding frame-rate of the DSRA is much higher than that of the S2 in encoder-II, as shown in Table 7.

On implementing a "real-time" software-based coder, the coding speed may be more important than the generated bit-rate. It will be a good deal to sacrifice several bit-rates for speeding up the coder. In summary, the implementation complexity of the DSRA is much lower than that of the S2 and is very close to that of the TSS; moreover, the DSRA does not need a large buffer to save the previously estimated motion vectors as the S2 does. Because the system complexity of the DSRA is much lower, the DSRA is easy to be implemented in hardware and is also easy to be optimized by the compiler while implemented in software.

4 CONCLUSION

Because the rapid progress of the IC technologies, it becomes possible to implement a real-time decoding system on a general-purpose processor. For most of the multimedia coding systems, where the complexity of an encoder is always higher than that of a decoder, e.g. in the MPEG-1 coding, this explains why it is still very difficult to implement a pure-software real-time encoding system. In order to speed up an encoding system, a very efficient block matching algorithm (with dynamically adjustable search range) is proposed which takes advantages of the correlation of motion vectors between neighboring blocks. It is shown by simulations that the proposed algorithm can reduce the complexity of a standard video encoding system. The average bit-rate of the coder using the proposed algorithm is also lower than the coder using some traditional heuristic search methods. It is believed that the proposed method can be applied to improve the efficiency of an encoding system and make the software-based real-time video encoding more feasible.

References

[1] *ISO/IEC 11172:1993, Information technology - coding of moving pictures, and associated audio for digital storage media up to about 1.5 Mbit/s.*

[2] F. A. Kamangar and K. R. Rao, "Fast algorithm for the 2-D discrete cosine transform," *IEEE Trans. Computers*, Vol. 31, No. 9, Sep. 1982, pp. 899–906.

[3] P. Duhamel and C. Gulillemot, "Polynomial transform for computing the 2-D DCT," *in Proc.*

ICASSP-90 (Albuquerque, NM), 1990, pp. 1515–1518.

[4] N. I. Cho and S. U. Lee, "Fast algorithms and implementations of 2-D discrete cosine transform," *IEEE Trans. Circuits and Systems*, Vol. 38, No. 3, Mar. 1991, pp. 297–305.

[5] E. Feig and S. Winograd, "Fast algorithms for the discrete cosine transform," *IEEE Trans. Signal Processing*, Vol. 40, No. 9, Sep. 1992, pp. 2174–2193.

[6] J. Chou, "The future of visual computing using an Alpha microprocessor—new motion video instruction," in *Proceedings of Multimedia Technology*, Taipei, Taiwan, Dec. 1996, pp. 205–216.

[7] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, Vol. COM-29, Dec. 1981, pp. 1799–1808.

[8] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," *in Proc. Nat. Telecommun. Conf.*, New Orleans, LA, Nov. 1981, pp. 5.3.1–5.3.5.

[9] S. Kappagantula and K. R. Rao, "Motion compensated interframe image prediction," *IEEE Trans. Commun.*, Vol. 33, No. 9, Sep. 1985, pp. 1011–1015.

[10] R. Srinivasan and K. R. Rao, "Predictive coding based on efficient motion estimation," *IEEE Trans. Commun.*, Vol. COM-33, Aug. 1985, pp. 888–895.

[11] A. Puri, H. M. Hang, and D. L. Schiling, "An efficient block-matching algorithm for motion compensated coding," *in Proc. IEEE ICASSP'87*, 1987, pp. 25.4.1–25.4.4.

[12] M. Ghanbari, "The cross-search algorithm for motion estimation," *IEEE Trans. Commun.*, Vol. COM-38, July 1990, pp. 950–953.

[13] C.-H. Hsieh, P.-C. Lu, J.-S. Shyn, E.-H. Lu, "Motion estimation algorithm using interblock correlation," *Electronic Letters*, Vol. 26, No. 5, March 1990, pp. 276–277.

[14] S. Zafar, Y.-Q Zhang, and S. Baras, "Predictive block-matching motion estimation for TV coding — part I: inter-block prediction," *IEEE Trans. Broadcasting*, Vol. 37, No. 3, Sep. 1991, pp. 97–101.

[15] J. Chalidabhongse and C.-C. Jay Kuo, "Fast motion vector estimation using multiresolution-spatio-temporal correlations," *to appear in IEEE Trans. Circuits and Systems for Video Technology.*

[16] L.-K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 6, no. 4, Aug. 1996, pp. 419–422.

[17] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 3, No. 2, April 1993, pp. 148–157.

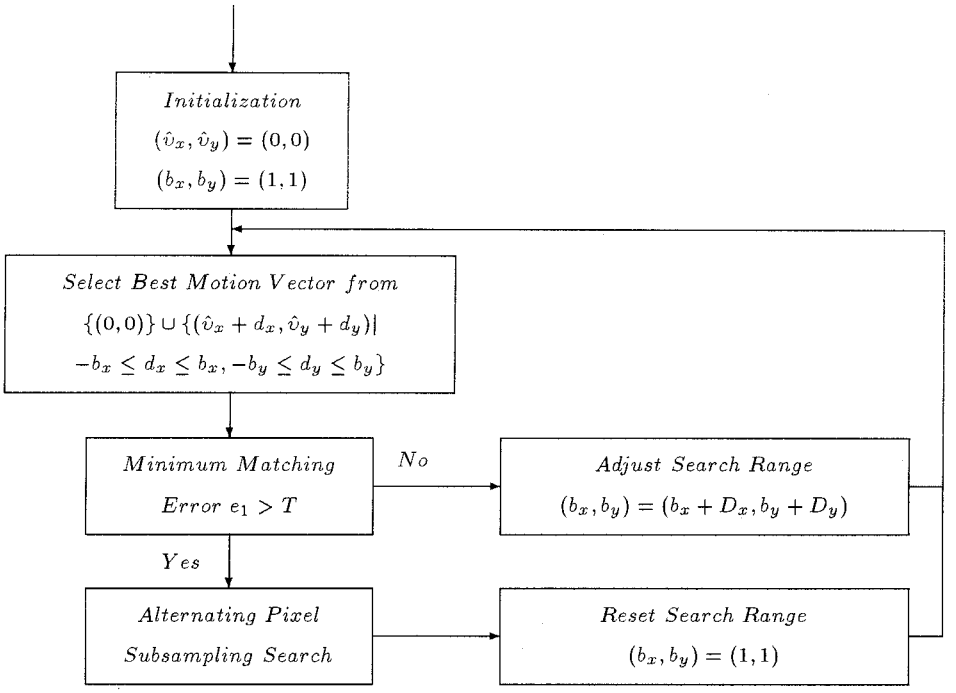


Figure 1: *The block diagram of the DSRA.*

Sequence Name	Resolution	# of Frames	Picture Characteristic
Salesman	352 × 288	149	head-and-shoulder
Miss America	352 × 288	150	head-and-shoulder
Football	352 × 240	125	violent motion
Flower Garden	352 × 240	115	pan
Mobile Calendar	352 × 240	140	pan
Table Tennis	352 × 240	112	zoom and scene change

Table 1: *The basic information of the test video sequences.*

	sales	miss	football	garden	mobile	tennis	average
FSA	984.92	984.92	973.70	973.70	973.70	973.70	977.44
TSS	30.62	30.91	30.44	30.59	30.38	30.46	30.57
S2	11.00	23.68	35.52	20.43	12.49	11.74	19.14
DSRA	10.42	18.76	34.74	17.83	10.47	10.39	17.10

Table 2: *The average block matching numbers (BMN) of different algorithms.*

	sales	miss	football	garden	mobile	tennis	average
TSS	3.11	3.14	3.13	3.14	3.12	3.13	3.13
S2	1.12	2.40	3.65	2.10	1.28	1.21	1.96
DSRA	1.06	1.90	3.57	1.83	1.08	1.07	1.75

Table 3: *The ratios (%) of the required block matching numbers for different fast BMAs to that for the FSA: BMN of fast algorithm / BMN of FSA.*

	sales	miss	football	garden	mobile	tennis	average
FSA	36.44	37.83	23.98	23.73	22.42	32.10	29.42
TSS	36.21	37.50	23.17	21.59	21.71	31.10	28.55
S2	36.12	36.87	22.65	23.13	21.44	28.83	28.17
DSRA	36.40	37.73	22.07	23.01	22.30	28.98	28.42

Table 4: *The average PSNR values of the estimated frames.*

	sales	miss	football	garden	mobile	tennis	average
TSS	30.75	30.99	28.57	30.11	35.63	28.61	30.78
S2	132.77	55.10	59.96	119.02	156	160.58	113.91
DSRA	157.89	63.99	59.23	102.36	160.33	237.05	130.14

Table 5: *Comparisons of the speed-ups to the FSA for different methods.*

	sales	miss	football	garden	mobile	tennis	average
FSA	0.39	0.28	0.17	0.18	0.18	0.24	0.24
TSS	4.38	3.22	2.32	2.37	2.31	3.04	2.94
S2	4.76	3.97	2.78	3.16	3.17	3.61	3.58
DSRA	5.96	3.69	2.94	3.20	3.36	4.10	3.88

(a)

	sales	miss	football	garden	mobile	tennis	average
FSA	1.67	2.91	0.81	0.83	0.79	1.03	1.34
TSS	16.69	23.50	9.59	9.54	9.52	12.03	13.48
S2	15.53	20.23	10.88	11.47	11.26	12.94	13.72
DSRA	23.16	28.05	12.78	13.19	13.92	16.15	17.88

(b)

Table 6: *Comparisons of the frame rates (frame/sec) of the MPEG encoder-I using different BMAs: (a) CIF, (b) QCIF.*

	sales	miss	football	garden	mobile	tennis	average
FSA	0.40	0.47	0.42	0.44	0.42	0.44	0.43
TSS	2.37	2.94	2.22	2.28	2.11	2.44	2.39
S2	3.81	4.86	2.54	3.52	3.43	3.23	3.57
DSRA	4.48	5.65	4.75	4.75	4.49	5.14	4.88

(a)

	sales	miss	football	garden	mobile	tennis	average
FSA	1.57	1.79	1.74	1.73	1.74	1.83	1.73
TSS	8.69	10.16	9.12	8.91	8.79	10.45	8.77
S2	11.43	12.14	8.90	10.05	10.79	10.85	10.69
DSRA	17.84	20.19	17.99	17.42	17.11	19.28	18.31

(b)

Table 7: *Comparisons of the frame rates (frame/sec) of the MPEG encoder-II using different BMAs: (a) CIF, (b) QCIF.*

	sales	miss	football	garden	mobile	tennis	average
FSA	0.1098	0.0632	0.4937	0.5985	0.6103	0.2308	0.3511
TSS	0.1129	0.0678	0.5404	0.7223	0.7468	0.2695	0.4100
S2	0.1111	0.0616	0.5081	0.6066	0.6134	0.2471	0.3580
DSRA	0.1128	0.0660	0.5470	0.6542	0.6423	0.2543	0.3794

(a)

	sales	miss	football	garden	mobile	tennis	average
FSA	0.1474	0.0570	0.8889	0.9714	1.1540	0.3609	0.5966
TSS	0.1480	0.0584	0.9078	1.0470	1.1835	0.3914	0.6227
S2	0.1478	0.0690	0.8937	0.9778	1.1575	0.3752	0.6035
DSRA	0.1494	0.0571	0.9382	1.0270	1.1808	0.3791	0.6219

(b)

Table 8: *The average bit-rates of the MPEG encoder-I using different BMAs: (a) CIF, (b) QCIF.*

	sales	miss	football	garden	mobile	tennis	average
FSA	0.1412	0.0892	0.6763	0.9151	1.2749	0.3756	0.5787
TSS	0.1418	0.0891	0.6860	0.9274	1.2852	0.3862	0.5860
S2	0.1426	0.0852	0.8184	1.4861	1.4897	0.4355	0.7429
DSRA	0.1476	0.0933	0.8364	1.6181	1.6272	0.4365	0.7932

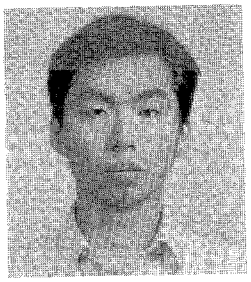
(a)

	sales	miss	football	garden	mobile	tennis	average
FSA	0.3247	0.1487	1.0388	1.3260	1.8604	0.5608	0.8766
TSS	0.3250	0.1485	1.0418	1.3346	1.8708	0.5646	0.8809
S2	0.3261	0.1688	1.1313	1.7678	1.9741	0.6180	0.9977
DSRA	0.3302	0.1536	1.1462	1.9067	2.0520	0.6118	1.0334

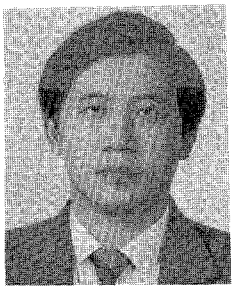
(b)

Table 9: *The average bit-rates of the MPEG encoder-II using different BMAs: (a) CIF, (b) QCIF.*

Biographies



Chun-Hung Lin was born in Kaohsiung, Taiwan, in 1968. He received the B.S. degree in information engineering from Tatung Institute of Technology, Taiwan, R.O.C. in 1991, and the M.S. degree in computer science and information engineering from National Taiwan University (NTU), Taipei, Taiwan. in 1993. He is currently a Ph.D. student in the Dept. of Computer Science and Information Engineering, NTU. His current research interests include image and video coding, multimedia systems, and genetic algorithms.

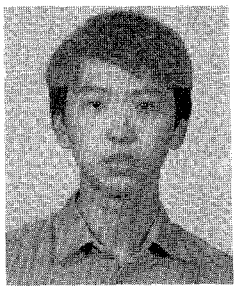


Ja-Ling Wu received the B.S. degree in electronic engineering from Tamkang University, Tamshoei, Taiwan, R.O.C. in 1979, and the M.S. and Ph.D. degree in electrical engineering from Tatung Institute of Technology Taipei, Taiwan, in 1981 and 1986, respectively.

From 1986 to 1987, he was an Associate professor of the Electrical Engineering Department at Tatung Institute of Technology, Taipei, Taiwan. Since 1987, he has been with the Department of Computer Science and Information Engineering, NTU, where he is presently a Professor. He is also the Head of the Department of Information Engineering, National Chi Nan University, Puli, Taiwan.

Dr. Wu was the recipient of the 1989 Outstanding Youth Medal of the Republic of China and the Outstanding Research Award sponsored by the National Science Council, from 1987 to 1992.

Dr. Wu has published more than 100 technique and conference papers. His research interests include neural networks, VLSI signal processing, parallel processing, image coding, algorithm design for DSP, data compression, and multimedia systems.



Yi-Shin Tung received the B.S. degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan. in 1996. He is currently a M.S. student in the Dept. of Computer Science and Information Engineering, National Taiwan University. His current research interests include image and video coding, multimedia systems.