論文
5-1-15

A New Fast Motion Estimation Algorithm Based on Block Sum Pyramid Algorithm

정수목(Soo-Mok Jung)[1)]

요 약

본 논문에서는 블록 합 피라미드 알고리즘에 기초한 빠른 움직임 추정 알고리즘을 제안하였다. Efficient Multi-level Successive Elimination Algorithm의 Spiral Diamond Mesh Search 기법과 Partial Distortion Elimination 기법을 개선하여 이들을 블록 합 피라미드 알고리즘에 적용하였다. 제안된 알고리즘의 움직임 추정 정확도는 거의 100%이고 블록 합 피라미드 알고리즘의 연산량을 효과적으로 줄였다. 실험을 통하여 제안된 알고리즘의 효율성을 확인하였다.

ABSTRACT

In this paper, a new fast motion estimation algorithm which is based on the Block Sum Pyramid Algorithm(BSPA) is presented. The Spiral Diamond Mesh Search scheme and Partial Distortion Elimination scheme of Efficient Multi-level Successive Elimination Algorithm were improved and then the improved schemes were applied to the BSPA. The motion estimation accuracy of the proposed algorithm is nearly 100% and the cost of Block Sum Pyramid Algorithm was reduced in the proposed algorithm. The efficiency of the proposed algorithm was verified by experimental results.

Key words: Block Sum Pyramid Algorithm, Block Matching, Motion estimation, Motion vector.

논문접수 : 2004. 1. 7.
심사완료 : 2004. 1. 15.

1) 정회원 : 삼육대학교 컴퓨터공학과 부교수

1.Introduction

Motion estimation, which has been widely used in various image sequence coding schemes, plays a key role in the transmission and storage of video signals at reduced bit rates. There are two classes of motion estimation methods, block matching algorithms (BMA)[1][2] and pel-recursive algorithms (PRA)[3]. The accuracy and efficiency of motion estimation affects the efficiency of temporal redundancy removal. Due to their implementation simplicity, block matching algorithms have been widely adopted by various video coding standards such as CCITT H.261[4], ITU-T H.263[5], and MPEG[6]. In BMA, the current image frame is partitioned into fixed-size rectangular blocks. The motion vector for each block is estimated by finding the best matching block of pixels within the search window in the previous frame according to matching criteria.

Although Full Search algorithm (FSA) finds the optimal motion vector by searching exhaustively for the best matching block within the search window, its high computation cost limits its practical applications. To reduce computation cost of FSA, many fast block matching algorithms such as three step search[7], 2-D log search[2], orthogonal search[8], cross search[9], one-dimensional full search[10], variation of three-step search [11][12], unrestricted center-biased diamond search[13] etc. have been developed. As described in [14], these algorithms rely on the assumption that the motion compensated residual error surface is a convex function of the displacement motion vectors, but this assumption is rarely true [15]. Therefore, the best match obtained by these fast algorithms is basically a local optimum.

Without this convexity assumption, Successive Elimination Algorithm(SEA)[16] proposed by Li and Salari reduces the computation cost of the FSA. To reduce the computation cost of SEA, Block Sum Pyramid Algorithm(BSPA)[17] and Multi-level Successive Elimination Algorithm(MSEA)[18] were proposed. Our research team proposed Efficient Multi-level Successive Elimination Algorithms for Block Matching Motion Estimation(EMSEA)[19] to reduce the computation cost of MSEA. In this proposed algorithm, the PDE and the Spiral Diamond Mesh Search Algorithm of EMSEA were improved and then the improved schemes were applied to BSPA. The motion estimation accuracy of the proposed algorithm is nearly 100% and the computation cost of BSPA is reduced by using this proposed algorithm.

2.Block Sum Pyramid Algorithm

The SEA achieves the same estimation accuracy as the FSA while requiring less computation time. In SEA, the displacement vector of the corresponding block in the previous frame is used as the initial motion vector for the present template block[23]. The SEA uses the sum norm of a block as a feature to eliminate unnecessary search points. The sum norm of a block B of size NxN is defined as

$$S_B = \sum_{i=1}^{N}\sum_{j=1}^{N} |B(i,j)| \qquad (1)$$

where $B(i, j)$ is the gray level of the (i, j)th pixel of block B. Let S_T be the sum norm of the template block T, S_X be the sum norm of a candidate matching block X, and *curr_MADmin* be the current minimal MAD during the search process. Let MAD(T, X) be the MAD between T and X and is defined as

$$\mathrm{MAD}(T, X) = \sum_{i=1}^{N}\sum_{j=1}^{N} |T(i,j) - X(i,j)| \qquad (2)$$

where $T(i, j)$ and $X(i, j)$ represent the gray values of the (i, j)th pixels of T and X. The authors had shown that the following inequality is true in [18]:

$$\mathrm{MAD}(T, X) \geq \sum_{i=1}^{N}\sum_{j=1}^{N} |S_T - S_X| \qquad (3)$$

Based on the above inequality, the SEA discards each candidate matching block X with $|S_T - S_X| \geq curr\_\mathrm{MAD}_{min}$, which can save a lot of search time.

Block Sum Pyramid Algorithm can eliminate those impossible matching blocks by exploiting the sum pyramid structure of a block. An image pyramid is a hierarchical data structure originally developed for image coding. Assume that each block is of size NxN with $N=2^n$. Then, for each block X, a pyramid of X can be defined as a sequence of blocks $\{ X^0, \ldots, X^{m-1}, X^m, X^{m+1}, \ldots, X^n\}$ with X^{m-1} having size 2^{m-1} x 2^{m-1} and being a reduced-resolution version of X^m as shown in <Fig. 1>. Note that X^0 has only one pixel. A pyramid data structure can be formed by successively operating over 2x2 neighboring pixels on the higher levels. That is, the value of a pixel $X^{m-1}(i, j)$ on level m-1 can be obtained from the values of the corresponding 2x2 neighboring pixels $X^m(2i-1, 2j-1)$, $X^m(2i-1, 2j)$, $X^m(2i, 2j-1)$, and $X^m(2i, 2j)$ on level m as shown in equation (4).

$$X^{m-1}(i, j) = X^m(2i-1, 2j-1) + X^m(2i-1, 2j) + X^m(2i, 2j-1) + X^m(2i, 2j)\} \qquad (4)$$

For two blocks X and Y, let $\mathrm{MAD}^m(X, Y)$ be $\mathrm{MAD}(X^m, Y^m)$, i.e.,

$$\mathrm{MAD}^m(X,Y) = \sum_{j=1}^{2^m}\sum_{h=1}^{2^m} |X^m(j,h) - Y^m(j,h)| \qquad (5)$$

where $X^m(j, h)$ and $Y^m(j, h)$ represent the values of the (j, h)th pixels on X^m and Y^m respectively. Thus, on the top level, $\mathrm{MAD}^0(X, Y)=|S_X - S_Y|$. From the above definition, the following theorem holds.

$$\mathrm{MAD}(X, Y) \geq \mathrm{MAD}^{n-1}(X, Y) \geq \mathrm{MAD}^{n-2}(X, Y) \geq \ldots \geq \mathrm{MAD}^0(X, Y) \qquad (6)$$

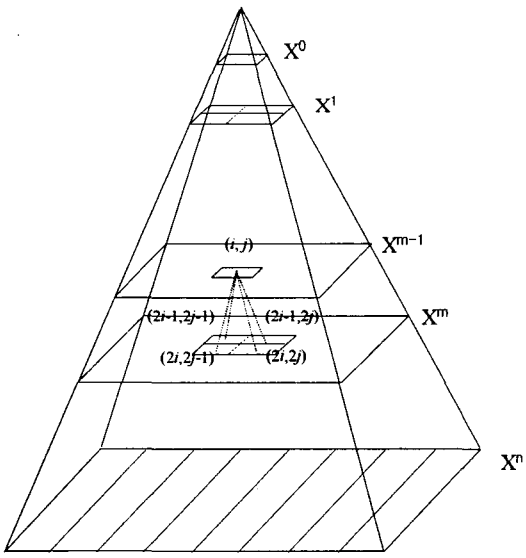
Block Sum Pyramid Algorithm uses the above theorem (6). The Block Sum Pyramid Algorithm first constructs the sum pyramid of every block that corresponds to a search position in the previous frame. To search for the best matching block of a template block T, the sum pyramid of T is established. Then, the MAD between T and the block with displacement vector (0, 0) is evaluated, and this value is considered as the current minimum MAD(symbolized as $curr\_\mathrm{MAD}_{min}$).

For any other search block X, the algorithm first checks the MAD on the top level, $\mathrm{MAD}^0(T, X)$. If $\mathrm{MAD}^0(T, X)$ is greater than $curr\_\mathrm{MAD}_{min}$, this block can be eliminated from being considered as the best matching block. Otherwise, the MAD on the first level is checked.

If $\mathrm{MAD}^1(T, X)$ is greater than $curr\_\mathrm{MAD}_{min}$, for the same reason as above, this block can be eliminated. If it is not, the second level is tested. The process is repeated until this block is eliminated or the bottom level is reached. If the bottom level is reached, then $\mathrm{MAD}(T, X)$ is calculated and checked. If $\mathrm{MAD}(T, X) \leq curr\_\mathrm{MAD}_{min}$, the current minimum distortion $curr\_\mathrm{MAD}_{min}$ is replaced with $\mathrm{MAD}(T, X)$. Block Sum Pyramid Algorithm can eliminate many

search blocks without evaluating their MADs. Assume that the size of the image frame is WxH. For each level of the pyramid, calculation of the sum of 2x2 neighboring pixels requires $3(W-1)(H-1)$ additions. However, using the idea for fast calculation of the sum norm developed in [18], the complexity can be reduced to be $(2W-1)(H-1)$ additions for each level. If the block size is 16x16, i.e., N=16, the overhead for constructing the sum pyramid is $4(2W-1)(H-1)$. Since there are $(W/N)(H/N)$ template blocks in an image frame, the computation overhead for each template block is

$$4(2W-1)(H-1)/[(W/N)(H/N)] = N^2(8 - 4/W - 8/H + 4/WH) \approx 8N^2 \quad (7)$$



<Fig. 1> A pyramid data structure

BSPA procedure is as follows:

step1 select an initial search block within the search window in the previous frame.

step2 calculate the motion vector (*MV*) and the MAD at the selected search block. these *MV* and MAD become the current temporary motion vector and the current minimum MAD respectively (*temp_MV=MV*, *curr*_MAD$_{min}$= MAD)

step3 select another search block among the rest of the search blocks

step 4.0 calculate the MAD^0(T, X) at the selected search block. if (*curr*_MAD$_{min}$ ≤ MAD^0(T, X) goto step 6

step 4.1 calculate the MAD^1(T, X) at the selected search block. if (*curr*_MAD$_{min}$ ≤ MAD^1(T, X)) goto step 6

.

.

.

step 4.(n-1) calculate the MAD$^{(n-1)}$(T, X) at the selected search block. if (*curr*_MAD$_{min}$≤MAD$^{(n-1)}$(T, X))

goto step 6

step 4.n calculate the MAD at the selected search block. if (*curr*_MAD$_{min}$≤MAD) goto step 6

step 5 *curr*_MAD$_{min}$=MAD.

calculate the motion vector at the selected search block. This motion vector becomes the current temporary motion vector (*temp_MV=MV*)

step 6 if (all the search blocks in the search window are not tested?) *goto* step3

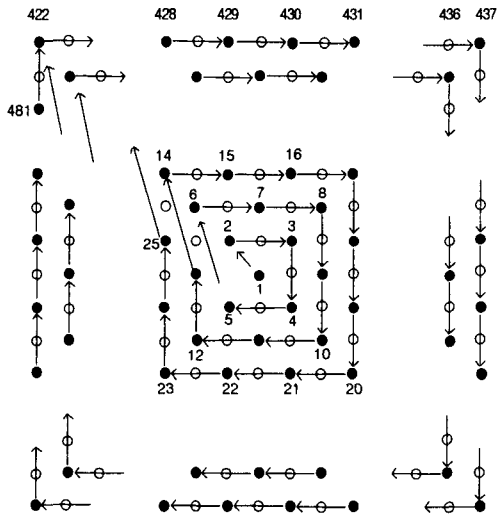
step 7 the optimum motion vector =*temp_MV*,

global minimum MAD=*curr*_MAD$_{min}$

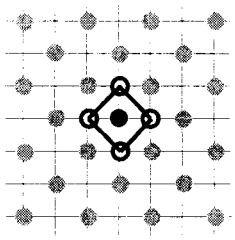
3. A New Fast Motion Estimation Algorithm

We improved the Spiral Diamond Mesh Search Algorithm and the PDE scheme of EMSEA[19] and then applied it to BSPA. In NFMEA, search points are sampled hierarchically as shown in <Fig. 2>. The maximum depth of hierarchical search points sampling is $L=N-1$ for the search windows with size $(2(N-1)+1,\ 2(N-1)+1)$ and the block with size NxN. The k-level NFMEA procedure is described in <Fig. 3>.

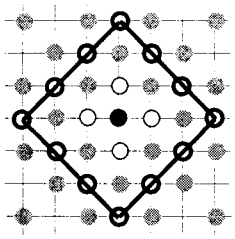
At 0-level, 481 black colored search points were sampled according to the spiral pattern from the origin to the points on the last outside square as shown in <Fig 2>(a). These black colored search points are distributed throughout the entire search window. BSPA is executed on the black colored search points form 1 to 481 sequentially. As shown in step 4 of NFMEA procedure, the black colored search points, which are supposed to have a motion vector, are pushed into a queue.



(a) 0-level



(b) 1-level



(c) 2-level

<Fig. 2> Hierarchical search points sampling

step2

1 select the origin point.

2 calculate the motion vector (***MV***) and the MAD at the selected search block. these ***MV*** and MAD become the current temporary motion vector and the current minimum MAD respectively (*temp_**MV***=***MV***, *curr*_MAD$_{min}$= MAD)

3 select next black colored search point along the spiral path at 0-level

4 execute the BSPA. At step 5 of the BSPA procedure, the coordinate of the selected search point is pushed into a stack.

5 if(all the black colored points at 0-level are not tested?) goto 3

6 while(the stack is not empty?){

7 pop the coordinate of a point from the stack

8.1 execute BSPA on four black circled points in 1-level surrounding the black colored search points which are supposed to have a motion vector

.

.

.

8.k. execute BSPA on $8(k-1)+4$ black circled points in k-level surrounding the black colored search points which are supposed to have a motion vector

9 }

10 the motion vector =*temp_**MV***,
minimum MAD=*curr*_MAD$_{min}$

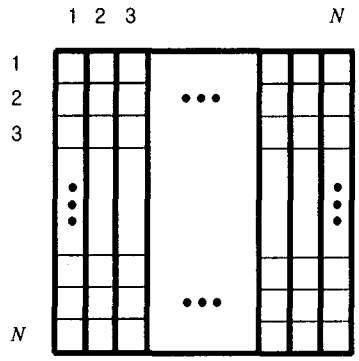
<Fig. 3> NFMEA procedure

At 1-level, BSPA is executed at the four black circled search points surrounding the black colored points which are supposed to have a motion vector in the 0-level search.
At the k-level, NFMEA searches $8(k-1)+4$ black circled search points surrounding the black colored search points which are supposed to have a motion vector in the 0-level search.
In NFMEA, by controlling the depth of hierarchical search point sampling, it is possible to adjust the real-time computational load and the motion estimation accuracy.
We let S be the average number of search points in the search window, C be the average number of operations per search point in the BSPA, D be the average depth of stack per frame, and V be the number of the motion vectors for 1 frame. In BSPA, SC operations are needed for motion estimation to find the motion vector of one target block. If we use k-level NFMEA, the operations can be reduced to approximately $(S/2+4k^2D/V)C$ operations.

Partial distortion elimination (PDE)[21] can be used in with BSPA to reduce the computation further for these vectors where MAD(T, X) must actually be computed. PDE is an effective speed up technique used in vector quantization to find a best reconstruction vector from a set of vector code words.

We observe that since all of the terms in the equation (2) are positive, if at any point the partially evaluated sum exceeds the current minimum MAD($curr\_MAD_{min}$), that candidate block X cannot be the best matching block and the remainder of the sum does not need to be calculated. While it is not efficient to test the partial sum against the $curr\_MAD_{min}$ every additional term is added, a reasonable compromise is to perform the test after N times additions when block size is NxN as shown in <Fig. 4>. So, the maximum number of PDE test is N.

In BSPA, MAD is calculated all the NxN pixels and then the calculated MAD is compared with $curr\_MAD_{min}$.



<Fig. 4> PDE test in MAD calculation

The PDE procedure is obtained by replacing step 4.n of BSPA with the following step.

step 4.n for PDE procedure:
calculate the MAD at the selected search block for N pixels.
if ($curr\_MAD_{min} \leq$ MAD) goto step6
else if MAD is not calculated for all pixels(NxN) then goto step 4.n

4. Experimental Results of NFMEA

The standard QCIF(176x144) video sequences in the frame work of H.263 such as "claire.qif", "salesman.qcif" were used in the experiment and we tested the first 50 frames of the video sequences. The block(Y component of the macro-block in H.263) size was 16x16 pixels (N=16). The size of the search window was 31x31

pixels (M=15) and only integer values for the motion vectors were considered. These experimental circumstances are equal to that of BSPA.

Experimental results are shown in table 1. In table 1, "m.e." means matching evaluation that require MAD calculation, avg. # of rows means the number of calculated rows in the MAD calculation before partial distortion elimination, NFMEA$_k$ means k-level NFMEA. Overhead(in rows) is the sum of all the computations except MAD calculation. "in rows" means that the computations are represented in order of 1 row MAD computations. "accuracy" means motion estimation accuracy.

Motion estimation accuracy is defined as a/t, where "t" is the total execution number of motion estimation and "a" is the accurate motion estimation number which is expressed in table 1. Accurate motion estimation finds the same motion vector of the full search algorithm. In the test, the sequences with qcif, motion estimations were executed 99 times per frame. The total number of motion vectors per frame is 4,950.

It is important to notice that with the NFMEA, the efficiency of the procedure depends on the order in which the candidate motion vectors are searched, and that the most promising candidates should be tested first. This eliminates the maximum number of candidates. In our experiment, we used spiral search.

The motion estimation accuracy of NFMEA was degraded negligibly as shown in <Table 1>. In k-level NFMEA, as k increases, the (avg. # of m.e./frame) increases and the motion estimation accuracy increases. The motion estimation accuracy of NFMEA is nearly 100%.

NFMEA can reduces the computations of BPSA by 1.7%, 10.5% with the motion estimation accuracy of BSPA for "claire.qif", "salesman.qcif" respectively.

<Table 1> The computations of NFMEA

video sequence	Algorithm	avg. # of m.e. /frame	avg. # of rows /m.e.	overhead (in rows)	total (in rows)	reduction (%)	accuracy
claire	BSPA	226.1	14.88	17,920.8	21,285.2		4,950
	NFMEA$_1$	172.1	14.95	10,331.9	12,904.8	39.4	4,935
	NFMEA$_2$	187.4	14.52	11,715.4	14,436.4	32.2	4,941
	NFMEA$_3$	199.8	13.90	12,845.5	15,622.7	26.6	4,945
	NFMEA$_4$	208.0	13.07	14,075.8	16,794.4	21.0	4,947
	NFMEA$_5$	214.3	11.87	15,474.2	18,017.9	15.3	4,949
	NFMEA$_6$	220.1	9.75	17,195.6	19,343.8	9.1	4,949
	NFMEA$_7$	223.5	7.27	19,310.9	20,931.3	1.7	4,950
salesman	BSPA	242.3	14.39	29,510.6	32,997.3		4,950
	NFMEA$_1$	228.8	14.23	16,619.4	19,875.2	39.8	4,941
	NFMEA$_2$	232.9	13.90	17,650.8	20,888.1	36.7	4,946
	NFMEA$_3$	236.4	13.53	19,526.1	22,724.6	31.1	4,948
	NFMEA$_4$	238.5	12.89	21,679.1	24,753.4	25.0	4,949
	NFMEA$_5$	239.2	11.87	24,103.2	26,942.5	18.3	4,949
	NFMEA$_6$	239.7	10.20	27,100.3	29,545.2	10.5	4,950
	NFMEA$_7$	239.8	8.28	30,371.8	32,357.3	1.9	4,950

5.Conclusions

A New Fast Motion Estimation Algorithm based on BSPA has been proposed to reduce the computations of BSPA. The Spiral Diamond Mesh Search Algorithm and the PDE scheme of EMSEA were improved and then the improved schemes were applied to BSPA in NFMEA.

The NFMEA outperforms BSPA and can reduce the computations of block matching calculation of BSPA 10.5% maximally with the motion estimation accuracy of BSPA. The NFMEA is a very efficient solution for video coding applications that require both very low bit-rates and good coding quality.

References

[1] H. G. Musmann, P. Pirsh, and H. J. Gilbert, "Advances in picture coding," *Proc. IEEE*, vol. 73, Apr. (1985), pp. 523-548.

[2] J. R. Jain, and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COMM-29, Dec. (1981), pp. 1799-1808.

[3] A. N. Netravali, and J. D. Robbins, "Motion compensated television coding: Part I," *Bell Syst. Tech. J*, vol. 58, Mar. (1979), pp. 631-670.

[4] CCITT Standard H.261, *Video codec for audiovisual services at px64 kbit/s*, ITU, 1990.

[5] ITU-T DRAFT Standard H.263, *Video coding for narrow telecommunication channel at (below) 64kbit/s*, ITU, Apr. 1995.

[6] ISO-IEC JTC1/SC2/WG11, *Preliminary text for MPEG video coding standard*, ISO, Aug. 1990.

[7] T. Koga, K. Iinuma, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," *Proc. Nat. Telecommunications Conf.*, Nov. (1981), pp. G5.3.1- G.5.3.5.

[8] A. Puri, H.-M. Hang, and D. L. Schilling, "An efficient block matching algorithm for motion compensated coding," *Proc. Int. Conf. Acoust., Speech, Signal Processing*, (1987), pp. 25.4.1-25.4.4.

[9] M. Ghanbari, "The cross-search algorithm for motion estimation," *IEEE Trans. Commun.*, vol. 38, no. 7, July (1990), pp. 950-953.

[10] M. J. Chen, L. G. Chen and T. D. Chiueh, "One-dimensional full search motion estimation algorithm for video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, Oct. (1994), pp. 504-509.

[11] H. M. Jong, L. G. Chen, and T. D. Chiueh, "Accuracy improvement and cost reduction of 3-step search block matching algorithm for video coding," *IEEE Trans. Circuits Syst. Video Technol*, vol. 4, Feb. (1994), pp. 88-91.

[12] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, Aug. (1994), pp. 438-442.

[13] J. Y. Tham, S. Ranganath, M. Ranganath, and A. Ali Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 4, Aug. (1998), pp. 369-377.

[14] B. Liu, and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. CircuitsSyst. Video Technol.*, vol. 3, no. 2, Apr. (1993), pp. 148-157.

[15] K. H. K. Chow, and M. L. Liou, "Genetic motion search algorithm for video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, Dec. (1993), pp. 440-446.

[16] W. Li, and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Trans. Image Processing*, vol. 4, No.1, Jan. (1995), pp. 105-107.

[17] C.H. Lee, and L.H. Chen, "A Fast Motion Estimation Algorithm Based on the Block Sum Pyramid Algorithm," IEEE Trans. Image Processing, Vol. 6, No.11, pp.1587-1591, Nov. 1997.

[18] X. Q. Gao, C.J. Duanmu, and C.R. Zou,

"A multilevel successive elimination algorithm for block matching motion estimation," *IEEE Trans. Image Processing*, Vol. 9, No.3, Mar. (2000), pp.501-504.

[19] S. M. Jung, S. C. Shin, H. Baik, and M. S. Park, "Efficient Multilevel Successive Elimination Algorithms for Block Matching Motion Estimation," *IEE Vision and Image Signal Processing*, vol.149, No.2, Apr. (2002), pp. 73-84.

[20] H.S. Wang and R. M. Mersreau, "Fast Algorithms for the Estimation of Motion Vectors," *IEEE Trans. Image Processing*, vol. 8, No.3, Mar. (1999), pp. 435-438.

[21] T. M. Apostol. *Mathematical Analysis*, Reading, MA: Addison-Wesley, 1975.

[22] H. S. Wang and R. M. Mersereau, "Fast Allgorithms for the Estimation of Motion Vectors," *IEEE Trans. Image Processing*, Vol. 8, No.3, Mar. (1999), pp.435438.

[23] Y. Q. Zhang and S. Zafar, "Predictive Block-matching Motion Estimation for TV Coding-Part II: Interframe Prediction," *IEEE Trans. Broadcast.*, vol. 37, Sept. (1991), pp. 102-105.

[24] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*, Boston, MA: Kluwer, 1991.

정수목

1984 경북대학교 전자공학과
(학사)
1986 경북대 대학원 전자공학과
(석사)
1986~1991 LG 정보 통신
연구소 연구원
2002 고려대 대학원
컴퓨터학과(박사)
1998~현재 삼육대학교 컴퓨터학과 부교수
관심분야: 멀티미디어, 컴퓨터시스템