A Scalable Pattern Omission Method of Deblocking Filter for H.264/AVC Baseline Profile

Jae Gon Kim, Jin Woo Hwang and Jun-Dong Cho
Sung Kyun Kwan University
300 Cheoncheon-dong, Jangan-gu, Suwon, Gyeonggi-do 440-746, Korea
Tel: +82-31-290-7200, Fax: +82-31-290-7683
E-mail: gon9031@skku.edu

Abstract: In this paper, we present the so-called scalable pattern omission method for a deblocking filter. This exploits specific characteristics of the human visual system. The baseline of our method is to reduce processing time of the deblocking filter. In the development of the H.264/AVC decoder, the processing Deblocking filter consumes the most computation resource. We proposed a scalable pattern omission method during filtering to decrease macroblocks that are filtered by the deblocking filter. Simulation result shows that our method can achieved performance enhancement ranging about 20% to 70% in the processing time.

1. Introduction

H.264/AVC or MPEG4 part 10 is developed by the Video Coding Experts Group (VCEG) of ITU-T and the Moving Picture Experts Group (MPEG) of ISO/IEC [1]. Comparing with the existing video compression standard like H.263 or MPEG-2, H.264/AVC provides better coding efficiency than previous coding standards does. With 4x4 block size that is smaller than block size of other previous standard, integer discrete cosine transform, and adaptive in-loop deblocking filter, high coding efficiency could be achieved [2]. The main features of H.264/AVC are 1) introduction of advanced motion estimation, 2) improvement of entropy coding such as CAVLC (Content Adaptive Variable Length Coding) and CABAC (Content Adaptive Binary Arithmetic Coding), 3) advancement of Deblocking filter [3]. The deblocking filter reduces blocking artifacts that are produced by block-based video coding. In the previous version, the deblocking filter was not standardized. In H.264, the deblocking filter is positioned in a prediction loop to increase coding efficiency and improve video quality. However, the deblocking filter increases computational complexity in encoding and decoding.

In this study, we propose a scalable pattern omission method for the deblocking filter. This exploits specific characteristics of the human visual system. In general, humans concentrate on the center of screen when they watch a video and show an optical illusion. Research [4] shows that humans usually disregard outlines of an image frame in a video sequences. We designed a filtering method using this characteristic. Our method saves deblocking time by macroblocks omission in the picture outline.

The structure of this paper is as follows. Section 2 describes a pattern omission method for the deblocking filter and introduces various patterns. Section 3 elaborates on a simulation method to prove our idea and shows the result of this research. Finally, the conclusions and future work are presented in Section 4.

2. Pattern omission method

2. 1 Method concept

A person’s eye has one focus. When people stare at a sequence of image, their relative focus is sensitive to the center of the image rather than the periphery. Based on this fact, we attempted to skip deblocking filtering in outline of a frame or between frames.

Figure 1 shows a basic raster scan order, which is recommended to read macroblocks in a frame by JM software. In this case, we try to skip outline of a frame. Figure 2 shows new raster scan order. Level means a number of boundaries. In this example, square level 1 represents that we will skip one of boundary and square level 2 will skip two of boundaries. Originally, deblocking filter will filter 99 macroblocks when image sequence is QCIF format. However, in our proposal, square level 1 will filter 63 macroblocks, and square level 2 will filter only 35 macroblocks. The processing ratio of pattern omission method is 63% (level 1) and 35% (level 2), respectively, compared with JM software method. Also, we realized that our method decreases processing time dramatically.

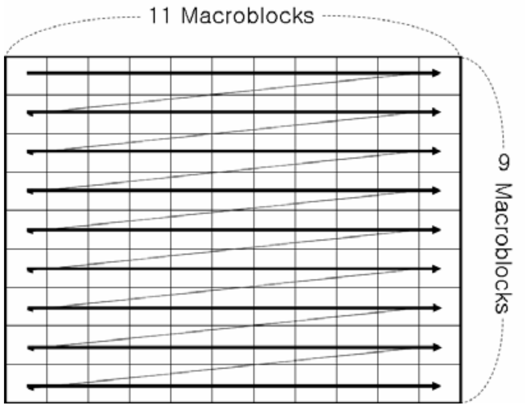


Figure 1. A basic raster scan order. QCIF format (11 x 9 Macroblocks, Macroblock = 16 x 16 pixels)

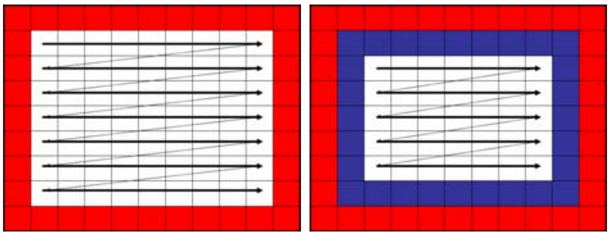
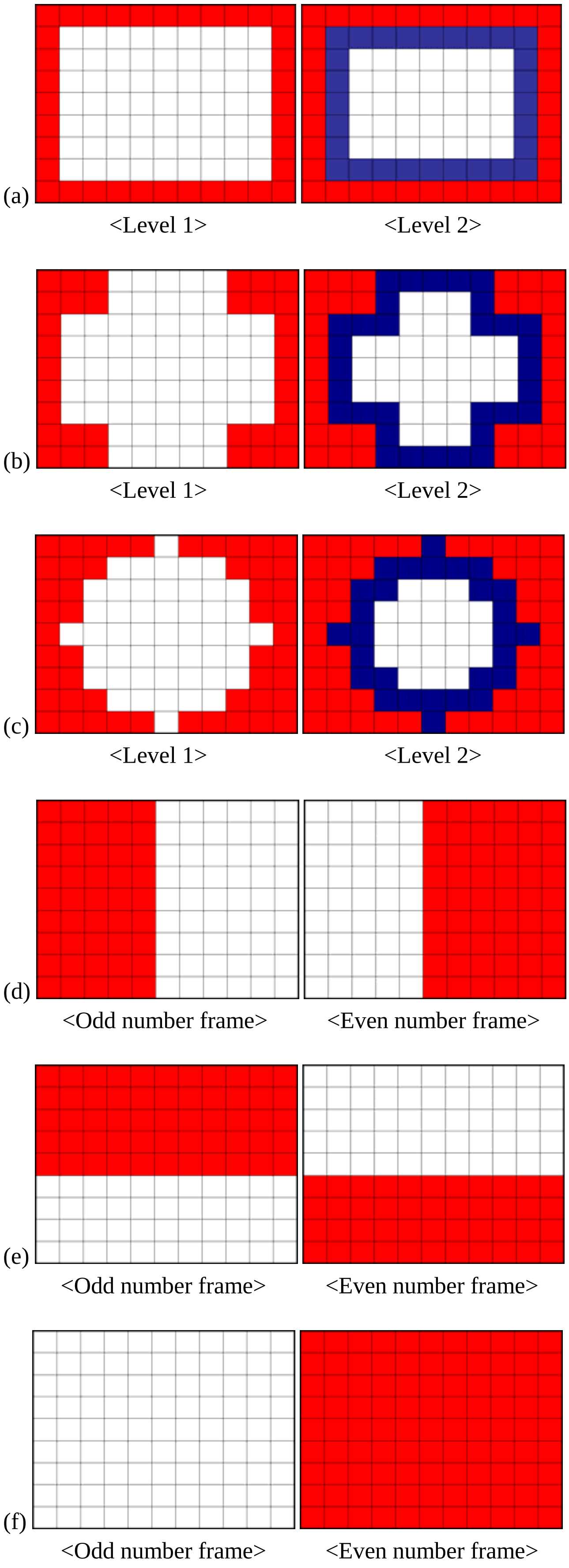


Figure 2. A raster scan order of pattern omission method. QCIF format (left: square level 1, right: square level 2)

2. 2 Vairous omission patterns



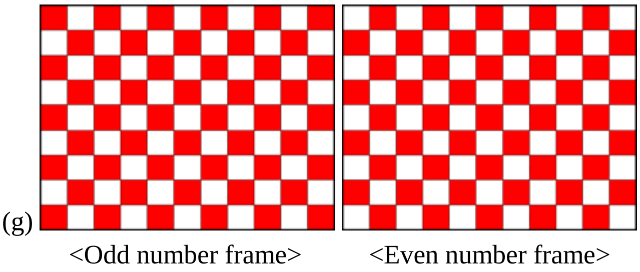


Figure 3. Various test patterns. QCIF format. Only white blocks will be filterd. (a) left: square level 1, right: square level 2 (b) left: Cross level 1, right: Cross level, (c) left: Circle level 1, right: Circle level 2, (d ~ g) Omission patterns during frame order, (d) half and half (left – right), (e) half and half (top – bottom), (f) one and one, (g) check

Figure 3 shows various omission test patterns. Test pattern type from (a) to (c) are designed by using specific characteristics of the human visual system: humans concentrate on the center of screen when they watch a video. Test pattern type from (d) to (g) are orignated by using specific characteristics of the human visual system: an optical illusion.

3. Simulation method and result

3. 1 Simulation method

The proposed skipping method is simulated on an Intel Pentium4 Processor 2.40GHz with 1024MB of memory. The simulation is run under Windows XP Operating System and compiled with Microsoft Visual C++ 8.0. In all of our experiments, six sequences (300 frames), i.e., foreman, coastguard, container, mobile, mother and daughter, news, and silent are used with QCIF format (176×144) and encoding to the baseline profile.

3. 2 Simulation result

Figure 4 shows the result of simulation such that we could recognize some difference among images. In the magnified image of each result, the difference was larger. Among the sequences of image which is filtered with two different methods, we identified that the quality of most sequences were almost of the same.

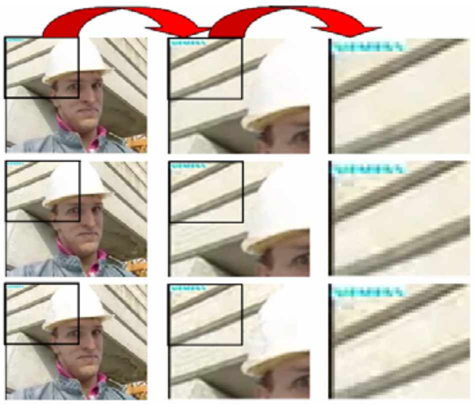
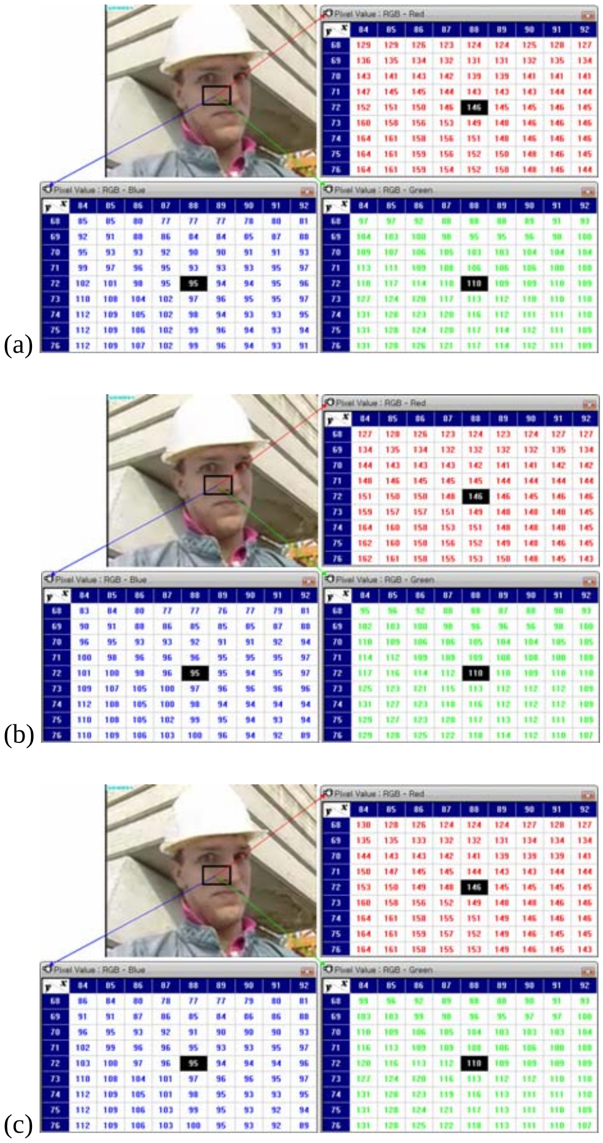


Figure 4. Result images of simulation. (top: JM software <reference>, middle: square level 1, bottom: square level 2)

Figure 5 shows the center area pixel value variation between images applying the different methods. Each image contains red, green, and blue components; three kinds of pixel values are exist. In Figure 5, we observed less difference among image pixel values. Comparing between JM software and square level 1, we discovered that 91% pixel values were of the same on the average. Comparing between JM software and square level 2, we discovered that 87% pixel values were of the same.



(a)

Pixel Value : RGB - Red

y x	84	85	86	87	88	89	90	91	92
68	129	129	126	123	124	124	125	128	127
69	136	135	134	132	131	131	132	135	134
70	143	141	143	142	139	139	141	141	141
71	147	145	145	144	143	143	143	144	144
72	152	151	150	146	146	145	145	146	145
73	160	158	156	153	149	148	146	146	146
74	164	161	158	156	151	148	146	146	146
75	164	161	159	156	152	150	148	146	145
76	164	161	159	154	152	150	148	146	144

Pixel Value : RGB - Blue

y x	84	85	86	87	88	89	90	91	92
68	85	85	80	77	77	77	78	80	81
69	92	91	88	86	84	84	85	87	88
70	95	93	93	92	90	90	91	91	93
71	99	97	96	95	93	93	93	95	97
72	102	101	98	95	95	94	94	95	96
73	110	108	104	102	97	96	95	95	97
74	112	109	105	102	98	94	93	93	95
75	112	109	106	102	99	96	94	93	94
76	112	109	107	102	99	96	94	93	91

Pixel Value : RGB - Green

y x	84	85	86	87	88	89	90	91	92
68	97	97	92	88	88	88	89	91	93
69	104	103	100	98	95	95	96	98	100
70	109	107	106	105	103	103	104	104	104
71	113	111	109	108	106	106	106	108	108
72	118	117	114	110	110	109	109	110	109
73	127	124	120	117	113	112	110	110	110
74	131	128	123	120	116	112	111	111	110
75	131	128	124	120	117	114	112	111	109
76	131	128	126	121	117	114	112	111	109

(b)

Pixel Value : RGB - Red

y x	84	85	86	87	88	89	90	91	92
68	127	128	126	123	124	123	124	127	127
69	134	135	134	132	132	132	132	135	134
70	144	143	143	143	142	141	141	142	142
71	148	146	145	145	145	144	144	144	144
72	151	150	150	148	146	146	145	146	146
73	159	157	157	151	149	148	148	148	145
74	164	160	158	153	151	148	148	148	145
75	162	160	158	156	152	149	148	146	145
76	162	161	158	155	153	150	148	145	143

Pixel Value : RGB - Blue

y x	84	85	86	87	88	89	90	91	92
68	83	84	80	77	77	76	77	79	81
69	90	91	88	86	85	85	85	87	88
70	96	95	93	93	92	91	91	92	94
71	100	98	96	96	96	95	95	95	97
72	101	100	98	96	95	95	94	95	97
73	109	107	105	100	97	96	96	96	96
74	112	108	105	100	98	94	94	94	94
75	110	108	105	102	99	95	94	93	94
76	110	109	106	103	100	96	94	92	89

Pixel Value : RGB - Green

y x	84	85	86	87	88	89	90	91	92
68	95	96	92	88	88	87	88	90	93
69	102	103	100	98	96	96	96	98	100
70	110	109	106	106	105	104	104	105	105
71	114	112	109	109	109	108	108	108	108
72	117	116	114	112	110	110	109	110	110
73	125	123	121	115	113	112	112	112	109
74	131	127	123	118	116	112	112	112	109
75	129	127	123	120	117	113	112	111	109
76	129	128	125	122	118	114	112	110	107

(c)

Pixel Value : RGB - Red

y x	84	85	86	87	88	89	90	91	92
68	130	128	126	124	124	124	127	128	127
69	135	135	133	132	132	131	134	134	134
70	144	143	143	142	141	139	139	139	141
71	150	147	145	145	144	143	143	144	144
72	153	150	149	148	146	145	145	145	145
73	160	158	156	152	149	148	148	146	146
74	164	161	158	155	151	149	146	146	146
75	164	161	159	157	152	149	146	145	145
76	164	161	158	155	153	149	146	145	143

Pixel Value : RGB - Blue

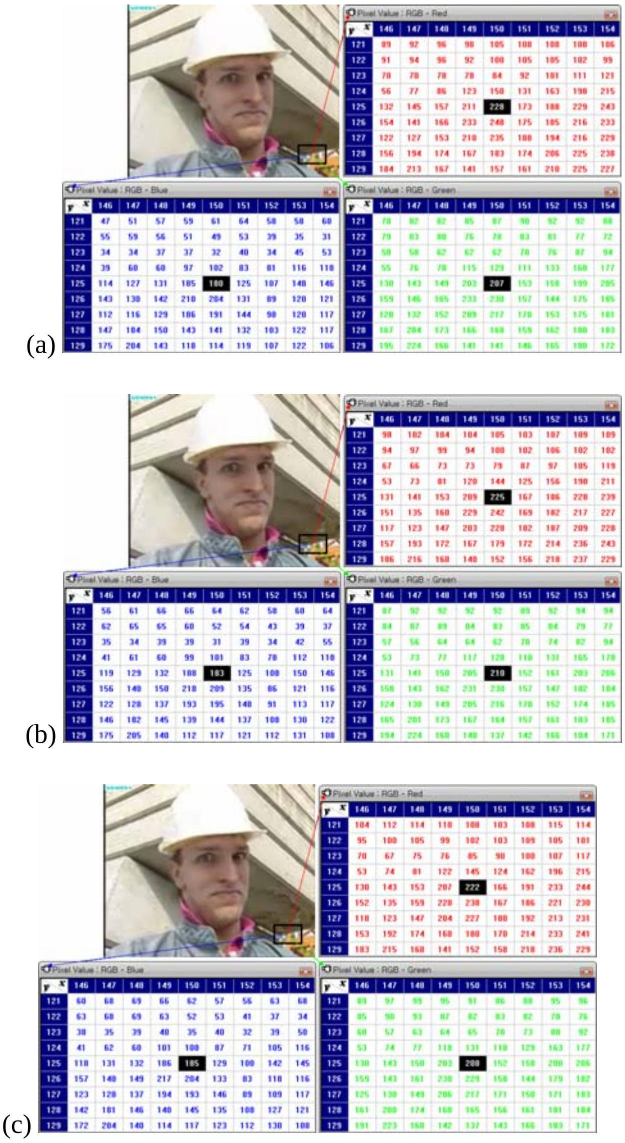
y x	84	85	86	87	88	89	90	91	92
68	86	84	80	78	77	77	79	80	81
69	91	91	87	86	85	84	86	86	88
70	96	95	93	92	91	90	90	90	93
71	102	99	96	96	95	93	93	95	97
72	103	100	97	96	95	94	94	94	96
73	110	108	104	101	97	96	96	95	97
74	112	109	105	101	98	95	93	93	95
75	112	109	106	103	99	95	93	92	94
76	112	109	106	103	100	95	93	92	89

Pixel Value : RGB - Green

y x	84	85	86	87	88	89	90	91	92
68	99	96	92	89	88	88	90	91	93
69	103	103	99	98	96	95	97	97	100
70	110	109	106	105	104	103	103	103	104
71	116	113	109	109	108	106	106	108	108
72	120	116	113	112	110	109	109	109	109
73	127	124	120	116	113	112	112	110	110
74	131	128	123	119	116	113	111	111	110
75	131	128	124	121	117	113	111	110	109
76	131	128	125	122	118	113	111	110	107

Figure 5. The variation of center area pixel values between images applying different methods. (a) JM software (b) Square level 1 (c) Square level 2

Figure 6 shows the boundary area pixel value's variation among images which was applied to different methods. In Figure 6, we observed a lot of difference among image pixel values. Comparing between JM software and square level 1, we discovered that 32% pixel values of the same were same. Comparing between JM software and square level 2, we discovered that 26% pixel values were of the same. However, when we simulated with new sequence image that shows less movement of action, we observed that boundary area of an image frame were 80% of the same in case of square level 1 and 44% of the same in case of square level 2.



(a)

Pixel Value : RGB - Red

y x	146	147	148	149	150	151	152	153	154
121	89	92	96	98	105	108	108	108	106
122	91	94	96	92	100	105	105	102	99
123	70	70	78	78	84	92	101	111	121
124	56	77	86	123	150	131	163	198	215
125	132	145	157	211	228	173	188	229	243
126	154	141	166	233	248	175	185	216	233
127	122	127	153	218	235	188	194	216	229
128	156	194	174	167	183	174	206	225	238
129	184	213	167	141	157	161	218	225	227

Pixel Value : RGB - Blue

y x	146	147	148	149	150	151	152	153	154
121	47	51	57	59	61	64	58	58	60
122	55	59	56	51	49	53	39	35	31
123	34	34	37	37	32	40	34	45	53
124	39	60	60	97	102	83	81	116	118
125	114	127	131	185	180	125	107	148	146
126	143	130	142	210	204	131	89	120	121
127	112	116	129	186	191	144	98	120	117
128	147	184	150	143	141	132	103	122	117
129	175	204	143	118	114	119	107	122	106

Pixel Value : RGB - Green

y x	146	147	148	149	150	151	152	153	154
121	78	82	82	85	87	90	92	92	88
122	79	83	80	76	78	83	81	77	72
123	58	58	62	67	62	70	76	87	94
124	55	76	78	115	129	111	133	168	177
125	130	143	149	203	207	153	158	199	205
126	159	146	165	233	230	157	144	175	185
127	128	132	152	209	217	178	153	175	181
128	167	204	173	166	168	159	162	188	183
129	195	224	166	141	141	146	165	190	172

(b)

Pixel Value : RGB - Red

y x	146	147	148	149	150	151	152	153	154
121	98	102	104	104	105	103	107	109	109
122	94	97	99	94	100	102	106	102	102
123	67	66	73	73	79	87	97	105	119
124	53	73	81	120	144	125	156	190	211
125	131	141	153	209	225	167	186	228	239
126	151	135	160	229	242	169	182	217	227
127	117	123	147	203	228	182	187	209	228
128	157	193	172	167	179	172	214	236	243
129	186	216	168	140	152	156	218	237	229

Pixel Value : RGB - Blue

y x	146	147	148	149	150	151	152	153	154
121	56	61	66	66	64	62	58	60	64
122	62	65	65	60	52	54	43	39	37
123	35	34	39	39	31	39	34	42	55
124	41	61	60	99	101	83	78	112	118
125	119	129	132	188	183	125	108	150	146
126	156	140	150	218	209	135	86	121	116
127	122	128	137	193	195	148	91	113	117
128	146	182	145	139	144	137	108	130	122
129	175	205	140	112	117	121	112	131	108

Pixel Value : RGB - Green

y x	146	147	148	149	150	151	152	153	154
121	87	92	92	92	92	89	92	94	94
122	84	87	89	84	83	85	84	79	77
123	57	56	64	64	62	70	74	82	94
124	53	73	77	117	128	118	131	165	178
125	131	141	150	205	210	152	161	203	206
126	158	143	162	231	230	157	147	182	184
127	124	130	149	205	216	170	152	174	185
128	165	201	173	167	164	157	161	183	185
129	194	224	168	140	137	142	166	184	171

(c)

Pixel Value : RGB - Red

y x	146	147	148	149	150	151	152	153	154
121	104	112	114	110	108	103	108	115	114
122	95	100	105	99	102	103	109	105	101
123	78	67	75	76	85	90	100	107	117
124	53	74	81	122	145	124	162	196	215
125	130	143	153	207	222	166	191	233	244
126	152	135	159	228	238	167	186	221	230
127	118	123	147	204	227	188	192	213	231
128	153	192	174	168	188	178	214	233	241
129	183	215	168	141	152	158	218	236	229

Pixel Value : RGB - Blue

y x	146	147	148	149	150	151	152	153	154
121	60	68	69	66	62	57	56	63	68
122	63	68	69	63	52	53	41	37	34
123	38	35	39	40	35	40	32	39	50
124	41	62	60	101	108	87	71	105	116
125	118	131	132	186	185	129	108	142	145
126	157	140	149	217	204	133	83	118	116
127	123	128	137	194	193	146	89	109	117
128	142	181	146	140	145	135	108	127	121
129	172	204	140	114	117	123	112	130	108

Pixel Value : RGB - Green

y x	146	147	148	149	150	151	152	153	154
121	89	97	99	95	91	86	88	95	96
122	85	90	93	87	82	83	82	78	76
123	60	57	63	64	65	70	73	80	92
124	53	74	77	118	131	118	129	163	177
125	130	143	150	203	208	152	158	200	206
126	159	143	161	230	229	158	144	179	182
127	125	130	149	206	217	171	158	171	183
128	161	208	174	168	165	156	161	181	184
129	191	223	168	142	137	143	166	183	171

Figure 6. The variation of boundary area pixel values between images applying different methods. (a) JM software (b) Square level 1 (c) Square level 2

Table 1. Comparisons between square level 1 compare to JM software.

Sequence Name	Speed up ratio	PSNR of JM reference	PSNR of Square Level 1	PSNR difference
Foreman	28.3 %	37.12 db	35.44 db	-1.68 db
Coastguard	31.7 %	34.77 db	33.49 db	-1.28 db
Container	34.9 %	36.17 db	35.92 db	-0.25 db
Mobile	37.8 %	34.22 db	33.25 db	-0.97 db
Mother and daughter	38.1 %	37.66 db	36.64 db	-1.02 db
News	23.5 %	37.13 db	36.95 db	-0.18 db
Silent	23.8 %	36.19 db	35.58 db	-0.61 db

Table 1 shows the result PSNR and speed up ratio of square level 1. In table 1, simulation result shows that square level 1 can achieved performance enhancement ranging about 20% to 40% in the processing time.

Table 2. Comparisons between square level 2 compare to JM software.

Sequence Name	Speed up ratio	PSNR of JM reference	PSNR of Square Level 1	PSNR difference
Foreman	59.2 %	37.12 db	34.82 db	-2.30 db
Coastguard	63.6 %	34.77 db	33.09 db	-1.68 db
Container	56.6 %	36.17 db	35.50 db	-0.67 db
Mobile	61.5 %	34.22 db	33.01 db	-1.21 db
Mother and daughter	65.3%	37.66 db	35.96 db	-1.70 db
News	50.5 %	37.13 db	36.12 db	-1.01 db
Silent	40.7 %	36.19 db	35.17 db	-1.02 db

Table 2 shows the result PSNR and speed up ratio of square level 2. In Table 1 and Table 2, we discovered some facts as follow: sequence images, which shown a lot of movement (i.e. foreman, coastguard, and mobile), showed a lot of PSNR loss, but gained much speed up rate. However, sequence images, which shown less movement of action (i.e. container, news, and silent), showed less PSNR loss, but gained a little speed up rate. This result can be used for decision that a number of boundaries will be skipped.

Table 3. Comparisons among various omission patterns compare to JM software (Used sequence frame: Foreman).

Pattern Name	Speed up ratio (level 1)	PSNR difference (level 1)	Speed up ratio (level 2)	PSNR difference (level 2)
Square	28.3 %	-1.68 db	59.2 %	-2.30 db
Cross	22.2 %	-1.95 db	52.3 %	-2.56 db
Circle	12.7 %	-1.82 db	26.5 %	-2.22 db
Half and half (left – right)	58.1 %	-2.64 db	None	None
Half and half (top – bottom)	76.4%	-2.62 db	None	None
One and one	50.9 %	-1.78 db	None	None
Check	50.8 %	-1.77 db	None	None

Table 3 shows the result PSNR and speed up ratio of arious omission patterns. From Table 3, we can determine that complex pattern, i.e. Cross and Circle, shows lower performance than simple pattern. There are no difference between one & one and check. In half and half cases, we expected that both will show similar result, but performance was quite different. Omission order of half and half (top-bottom) pattern follows raster scan order.

Figure 7 shows simulation result graph of various omission patterns in table 3.

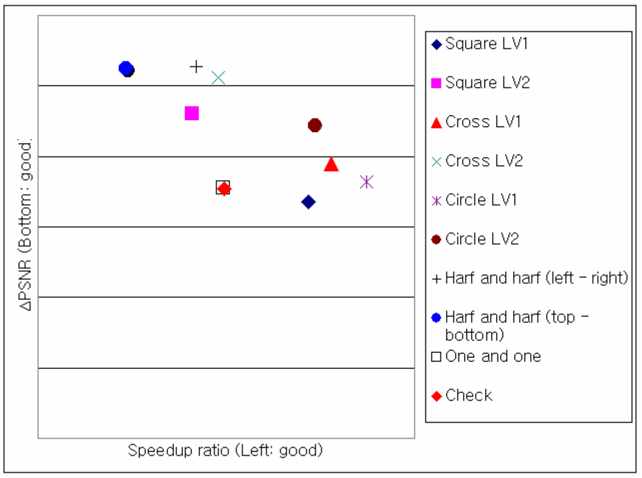


Figure 7. Simulation result graph of various omission patterns

In figure 7, we can find out that check pattern and one and one pattern are most useful patterns. Square level 1 pattern shows most good quality image. Half and half (top-bottom) pattern shows fastest processing time.

4. Conclusion

A scalable pattern omission method of deblocking filter is a new concept by using the specific characteristics of the human visual system. In this study we used QCIF format which is the smallest standard image format. However, actually, this method may apply large size of image format more efficiently. Through the simulation, we determined that check pattern can be most used in practical applications. Even though we used the smallest standard image format, we were able to show the feasibility of our method.

In the near future, we will extend this method to be implemented in architect hardware and elaborate a scalability of permission range for large size of image format.

Acknowledgments

This work was partially supported by "System IC 2010" project of Korea Ministry of Knowledge Economy, KOSEF, and IDEC.

References

[1] Joint Vido Team of ITU-T and ISO/IEC Recommendation H.264, “Advanced Video Coding for Generic Audiovisual Services,” ITU-T, March. 2005.

[2] Thomas, W. et al, “Overview of the H.264/AVC Video Coding Standard,” IEEE Trans. on Circuit and Systems for Video Technology, Vol. 13, No. 7, July. 2003, pp. 560-576.

[3] Thomas, W. et al, “Rate-constrained coder control and comparison of video coding standards,” IEEE Trans. on Circuit and Systems for Video Technology, Vol. 13, No. 7, July. 2003, pp. 688-703.

[4] Gwang-Hoon, P. et al, “Water Ring Scan Method for FGS Video Coding Schemes,” IEICE Trans. on Communications , Vol. E88-B, No. 2 , February. 2005, pp. 835-840.