Fast Overlapped Block Motion Estimation for Frame Rate Up Conversion

Yonghoon Kim and Jechang Jeong
Department of Electronic and Computer Engineering
Hanyang University
17 Haengdang-dong, Seongdong-gu, Seoul
SOUTH KOREA
hoonykim85@gmail, jjeong@hanyang.ac.kr

Abstract: - In this paper, we propose a fast motion estimation algorithm to be used for motion-compensated frame rate up conversion (FRUC). The FRUC techniques are used to enhance the visual quality of the low frame rate video on display. By using the motion vectors obtained from motion estimation process, the new frames can be interpolated. The motion estimation is the key part of the FRUC process but it is suffers from high computational complexity. Therefore we proposed fast overlapped block motion estimation algorithm which can be adapted to FRUC algorithm. We reduce the complexity of the motion estimation by using new sub-block calculation order based fast motion estimation algorithm.

Key-Words: - Frame rate up conversion, fast motion estimation, partial distortion search.

1 Introduction

Frame rate up-conversion refers to the technique that increase frame rate of the video from the one with a lower frame rate through the process of producing new frames and inserting them into the original video. FRUC has been used for the conversion between two display formats with different frame rates, or to remove the temporal redundancy in video coding. Recently, FRUC has become a new area of applications to the reduction of motion blur of the hold-type displays such as liquid crystal display (LCD). As the motion blur of the moving image on the LCD can be greatly reduced, major display manufacturers increase the frame rate of the TV from 60 Hz to 120 or 240 Hz.

Normally, motion compensated FRUC conducts two processes for interpolating a new frame. First part is motion estimation (ME). The ME process searches motion vectors by calculating the displacement of moving objects between two or more consecutives frames. Second one motion compensated interpolation (MCI) [1] which is a process to make an interpolated frame with motion vectors obtained from ME process.

Until now, several ME algorithms have been introduced and the most popular method is block matching algorithm (BMA) because of its superior scalability [2-4] and easy to implement in software and hardware. In FRUC, there are two motion estimation methods, bidirectional and unidirectional ME (UME). The bidirectional ME (BME) is an efficient method to reduce hoes and occluded region. Many BME algorithms are proposed such as extension of original BME [5], overlapped BME methods [6-8], and multi frame based BME [9]. To overcome the weakness of the UME method, Kang *et al*. proposed a FRUC algorithm using both BME and UME [10].

Most of the algorithms utilize at least one ME process to get the motion information. The ME process is the most computational process in FRUC because it searches all blocks in the search window. In addition many FRUC algorithms utilize overlapped block motion estimation (OBME) which improves the accuracy of the motion vector and it requires more computational complexity.

To reduce the computational burden, Many Fast ME algorithm have been introduces, such as successive elimination algorithm (SEA) [11], partial distortion search (PDS), normalized partial distortion search (NPDS) [12], and adjustable partial distortion search (APDS) [13]. Among them, APDS algorithm shows the fastest computation time and least image quality degradation. However, APDS is designed only for 16 x 16 block size and this algorithm cannot be adopted for OBME which has various block sizes. Therefore we propose new fast overlapped block motion estimation algorithm for variable block sizes.

2 Proposed algorithm

2.1 Bi-directional motion estimation

Table 1. Offsets of the *p*th partial distortion from the upper left corner pixel of a sub-block

k	(s_k, t_k)	k	(s_k, t_k)
1	(0,0)	9	(1,0)
2	(2,2)	10	(3,2)
3	(2,0)	11	(0,1)
4	(0,2)	12	(2,3)
5	(1,1)	13	(3,0)
6	(3,3)	14	(1,2)
7	(3,1)	15	(2,1)
8	(1,3)	16	(0,3)

Conventional BME searches a motion vector by using temporal symmetry between the corresponding blocks in the previous and current frames as shown in Fig. 1. When finding the best matching block, the BME uses the sum of bidirectional absolute differences (SBAD) which is given as follows:

$$SBAD(dx,dy) = \sum_{x\in W_x}\sum_{x\in W_y} \left| f_{n-1}(x-dx, y-dy) - f_n(x+dx, y+dy) \right| \quad (1)$$
$$v = \arg\min_{(dx,dy)\in S} \{SBAD(dx,dy)\}$$

where (*dx*,*dy*) denotes the candidate motion vector, and fn-1 and f_n are the previous and current frames. W_x and W_y denote the horizontal and vertical positions of a block respectively, and S represents the search window. From eq. (1), the final motion vector is selected when the candidate motion vector has the minimum SBAD value in the search window.

2.2 Proposed Fast Overlapped Block Motion Estimation (FOBME)

The original APDS is designed for only block which has 16 x 16 block size but the proposed algorithm expend this algorithm to 4M x 4M block size for the FRUC algorithm.

The partial distortion search (PDS) is terminated early when the calculated partial SAD d_k is greater than SAD_{min}. The PDS divides the macroblock into 4x4 sub-blocks, which are defined as

$$d_k = \sum_{i=i+4}^{N}\sum_{j=j+4}^{N} \left| f_{n-1}(x+i+s_k-dx, y+j+t_k-dy) - f_n(x+i+s_k+dx, y+j+t_k+dy) \right| , \quad (2)$$

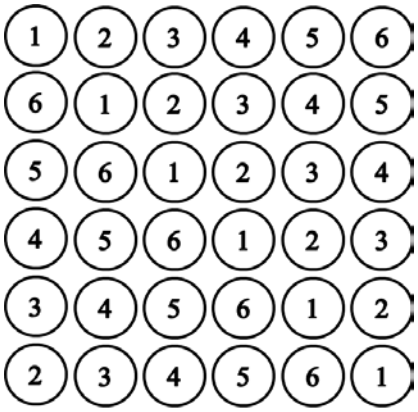


Fig. 1. Example of proposed computation order

$$D_k = \sum_{p=1}^{k} d_p \ , \quad (3)$$

where d_k is *k*th partial distortion, while (s_k, t_k) denotes the pixel coordinates of the upper left corner pixel of the sub-block which is shown in Table 1. During the block matching, the partial matching error D_k accumulated for every period, is computed and compared with the mini-mum SAD.

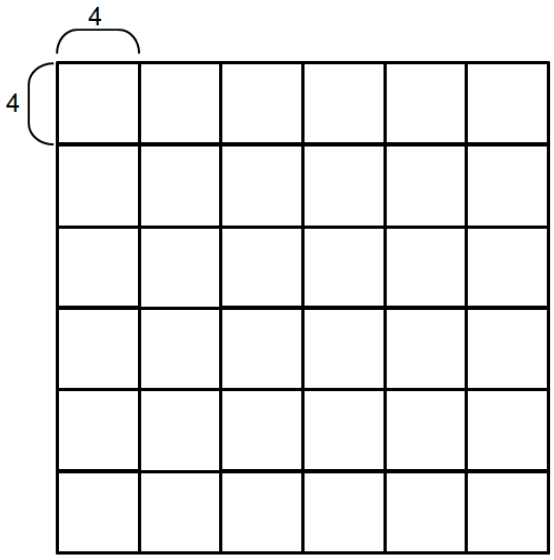
In order to reject the improper candidate earlier, the Normalized Partial Distortion Search (NPDS) uses normalization. Instead of using D_k, normalized partial distortion D_{norm} is compared with the normalized minimum SAD. D_{norm} is defined as follows:

$$D_{norm} = d_k \times \frac{N^2}{k} \ . \quad (4)$$

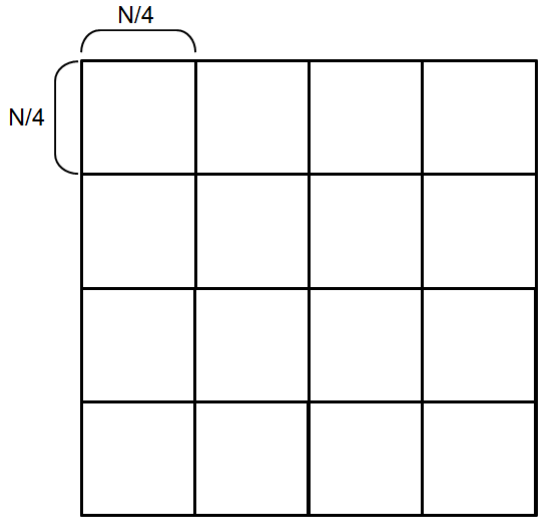
The probability of early rejection is greatly increased by adopting normalization. As a result, performance improves about from 10 to 12 times in terms of computation reduction; however, false rejection also occurs, because distortion does not have various motion vectors inside the block. Due to this erroneous rejection, the performance of the NPDS is not sufficient in comparison with that of FS.

The proposed extended APDS algorithm is combination of PDS and NPDS with quality factor *k*. It decides whether the block-matching on the current candidate continues or stops for the rest of the partial distortions. In this algorithm to achieve adjustable control, adjustable distortion *D(n,k)* is defined as

$$D(n,k) = ((1-k)\cdot n + k\cdot N^2)\cdot (SAD_{\min} / N^2) \quad (5)$$



(a)



(b)

Fig. 2. Two proposed block division methods (a) method A, (b) method B

where N^2 denotes the total number of pixels in the block and n denotes the number of counted pixels in partial distortion. SAD_{min} represents the minimum SAD value which is founded during motion estimation process. Using the SAD_{min} value, adjustable distortions are calculated for each step.

Based on sub block calculated order as shown in Fig.1, the partial distortion is calculated. Fig.1 shows an example of 24 x 24 block size using method A shown in Fig. 2(a) and this concept can be easily expended to block size which is 4M x 4M. If the calculated partial distortion is larger than the pre-calculated adjustable distortion, rest process of current candidate block is skipped and goes to the next block.

The proposed sub-block calculation can be used as two approaches. First one uses fixed 4x4 block method as shown in Fig. 2(a). In this case, we combine proposed block calculation order and existing sub block calculation order as shown in Table. 1. Second method is the inverse of first one which uses proposed block calculation order as sub block as shown in Fig. 2(b). The sub block size of second method can be determined as quarter of total block size. This sub block size does not contain overlapped block size.

3 Experimental result

The proposed algorithm is simulated using the 720p (1280×720): FourPeople (600 frames), Johnny (600 frames) KristenAndSara (600 frames), and 1080p (1920×1080) test sequences: BasketballDrive (500 frames), Cactus (500 frames), and ParkScene (240 frames). In this experiment, the proposed algorithm is tested on 2 different block sizes with ±16 search range. We evaluated the performance of the proposed in terms of the peak signal-to-noise ratio (PSNR) of the interpolated frames, which represents the metrics most commonly used for evaluating the image quality and processing time to check the computation reduction.

Table 2 and Table 3 show the experimental results of PSNR and computation time of 720p and 1080p video resolution, respectively. We used 16×16 block size and 4 overlapped area and it means 24×24 overlapped block size. Likewise, 32×32 block and 8 overlapped block size is 48×48 overlapped block size. We tested ten different k values from 0.01 to 0.1 to find the optimal value for different video resolution and block size. We compared two proposed block division method. We can find that when k value is 0.01, the results show the video quality for the 24×24 overlapped block size. For the 48×48 overlapped block size, 0.02 produces best image quality but 0.01 shows very small PSNR degradation than 0.02 with higher speed. From the results, the optimal k value can be chosen from 0.01 which satisfies both speed and video quality.

Table 4 and Table 5 show the result of proposed method B on 720p and 1080p videos, respectively. The method B also shows the best performance in term of both video quality and speed when k is 0.01. However, method B shows slightly better speed and similar video quality compared with method A when the block size is small in this experiment 24 x 24. When block size is 48 x 48, method B shows

Table 2. Experimental results using method A of PSNR (dB) and computation time (second) for 720p videos

sequence	*k*	BS:16 OL:4		BS:32 OL:8	
		PSNR	Time	PSNR	Time
Four People	FS	40.02	1652	40.35	1545
	0.01	40.16	58	40.37	27
	0.02	40.16	68	40.37	31
	0.03	40.16	77	40.37	36
	0.04	40.16	82	40.37	42
	0.05	40.16	93	40.37	48
	0.06	40.16	102	40.37	54
	0.07	40.15	110	40.37	59
	0.08	40.15	117	40.37	64
	0.09	40.15	124	40.37	69
	0.10	40.15	130	40.37	73
Johnny	FS	40.89	2069	41.43	1944
	0.01	41.25	87	41.63	43
	0.02	41.25	117	41.63	61
	0.03	41.24	146	41.62	81
	0.04	41.24	173	41.63	105
	0.05	41.24	203	41.63	129
	0.06	41.24	230	41.62	148
	0.07	41.24	247	41.62	162
	0.08	41.24	273	41.62	174
	0.09	41.24	291	41.62	187
	0.10	41.24	307	41.62	200
Kristen AndSara	FS	41.04	1851	41.49	1757
	0.01	41.16	76	41.47	34
	0.02	41.15	91	41.48	44
	0.03	41.15	98	41.48	56
	0.04	41.15	115	41.48	69
	0.05	41.15	132	41.48	82
	0.06	41.15	146	41.48	94
	0.07	41.15	158	41.48	104
	0.08	41.15	170	41.48	112
	0.09	41.15	182	41.48	120
	0.10	41.15	193	41.48	128

Table 3. Experimental results using method A of PSNR (dB) and computation time (second) for 1080p videos

sequence	*k*	BS:16 OL:4		BS:32 OL:8	
		PSNR	Time	PSNR	Time
Basket ballDrive	FS	28.37	3463	29.87	3368
	0.01	28.47	256	29.90	173
	0.02	28.47	355	29.90	250
	0.03	28.47	452	29.90	320
	0.04	28.46	528	29.90	386
	0.05	28.46	602	29.91	447
	0.06	28.46	676	29.91	504
	0.07	28.47	758	29.91	558
	0.08	28.47	815	29.91	608
	0.09	28.47	874	29.91	656
	0.10	28.47	908	29.91	698
Cactus	FS	30.35	3479	31.87	3392
	0.01	30.61	192	31.93	105
	0.02	30.58	245	31.93	145
	0.03	30.56	294	31.93	186
	0.04	30.55	341	31.93	224
	0.05	30.55	401	31.93	261
	0.06	30.55	438	31.93	297
	0.07	30.54	488	31.93	333
	0.08	30.54	511	31.93	368
	0.09	30.54	527	31.93	405
	0.10	30.54	562	31.93	436
Park Scene	FS	32.54	1643	34.69	1633
	0.01	32.85	86	34.72	56
	0.02	32.85	115	34.73	81
	0.03	32.83	143	34.73	104
	0.04	32.83	169	34.73	126
	0.05	32.83	194	34.73	147
	0.06	32.83	217	34.73	167
	0.07	32.82	239	34.73	187
	0.08	32.82	261	34.73	207
	0.09	32.82	283	34.73	227
	0.10	32.82	304	34.73	245

faster speed but produces slight quality degradation on 720p. For the 1080p videos, method B shows better image quality and speed on 24 x 24 and method A achieves better speed and quality on 48 x 48 than method B.

The proposed FOBME shows better video quality than full search algorithm and it has similar characteristic with APDS. The proposed method approximately 25 times faster than FS algorithm and it can be combined with other search point reduction algorithm to further reduce the computational complexity for FRUC using variable block size.

4 Conclusion

In this paper, we proposed fast overlapped block motion estimation algorithm for FRUC algorithm. We proposed a new block calculation order which can be expended any 4M x 4M size block motion estimation. The experimental result shows that the proposed algorithm approximately 25 times faster than full search algorithm when k is set as 0.01 and better video quality. The proposed algorithm can be combined search point reduction algorithm to further reduce the computational complexity.

Table 4. Experimental results using method B of PSNR (dB) and computation time (second) for 720p videos

sequence	*k*	BS:16 OL:4		BS:32 OL:8	
		PSNR	Time	PSNR	Time
Four People	FS	40.02	1652	40.35	1545
	0.01	40.19	46	40.36	22
	0.02	40.19	52	40.36	28
	0.03	40.20	62	40.36	33
	0.04	40.19	70	40.36	40
	0.05	40.19	80	40.36	48
	0.06	40.19	85	40.36	53
	0.07	40.19	91	40.36	59
	0.08	40.19	98	40.36	64
	0.09	40.19	104	40.36	68
	0.10	40.19	113	40.36	72
Johnny	FS	40.89	2069	41.43	1944
	0.01	41.24	69	41.58	41
	0.02	41.23	100	41.57	61
	0.03	41.23	125	41.56	79
	0.04	41.22	151	41.55	97
	0.05	41.21	174	41.54	114
	0.06	41.21	189	41.53	131
	0.07	41.20	210	41.53	149
	0.08	41.20	226	41.53	163
	0.09	41.19	247	41.53	182
	0.10	41.19	261	41.53	197
Kristen AndSara	FS	41.04	1851	41.49	1757
	0.01	41.15	56	41.46	30
	0.02	41.16	78	41.46	43
	0.03	41.15	93	41.47	54
	0.04	41.15	112	41.47	66
	0.05	41.14	126	41.47	76
	0.06	41.15	138	41.47	87
	0.07	41.14	149	41.47	97
	0.08	41.14	163	41.46	107
	0.09	41.14	173	41.46	116
	0.10	41.13	183	41.46	126

Table 5. Experimental results using method B of PSNR (dB) and computation time (second) for 1080p videos

sequence	*k*	BS:16 OL:4		BS:32 OL:8	
		PSNR	Time	PSNR	Time
Basket ballDrive	FS	28.37	3464	29.87	3368
	0.01	28.50	263	29.86	205
	0.02	28.51	360	29.85	297
	0.03	28.51	453	29.84	373
	0.04	28.50	542	29.83	445
	0.05	28.50	605	29.83	511
	0.06	28.49	672	29.82	578
	0.07	28.49	747	29.81	633
	0.08	28.49	808	29.80	687
	0.09	28.48	857	29.79	741
	0.10	28.48	915	29.79	794
Cactus	FS	30.35	3480	31.87	3392
	0.01	30.63	166	31.90	109
	0.02	30.62	242	31.89	164
	0.03	30.60	292	31.89	212
	0.04	30.59	350	31.88	256
	0.05	30.59	401	31.88	296
	0.06	30.58	448	31.88	335
	0.07	30.58	495	31.87	366
	0.08	30.57	545	31.87	393
	0.09	30.57	588	31.87	427
	0.10	30.57	626	31.87	459
Park Scene	FS	32.54	1643	34.69	1633
	0.01	32.91	86	34.66	58
	0.02	32.91	123	34.68	89
	0.03	32.90	158	34.67	115
	0.04	32.91	189	34.67	138
	0.05	32.90	219	34.67	160
	0.06	32.89	250	34.67	181
	0.07	32.89	270	34.67	201
	0.08	32.89	295	34.67	224
	0.09	32.89	318	34.67	240
	0.10	32.89	338	34.66	258

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT and future Planning(NRF-2015R1A2A2A01006004)

References:

[1] S.-J. Kang, D.-G. Yoo, S.-K. Lee, and Y. H. Kim, "Multiframe-based bilateral motion estimation with emphasis on stationary caption processing for frame rate up-conversion," *Consumer Electronics, IEEE Transactions on*, vol. 54, pp. 1830-1838, 2008.

[2] B.-D. Choi, J.-W. Han, C.-S. Kim, and S.-J. Ko, "Frame rate up-conversion using perspective transform," *Consumer Electronics, IEEE Transactions on*, vol. 52, pp. 975-982, 2006.

[3] G. De Haan, P. W. Biezen, H. Huijgen, and O. Ojo, "True-motion estimation with 3-D recursive search block matching," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 3, pp. 368-379, 388, 1993.
[4] K.-M. Yang, M.-T. Sun, and L. Wu, "A family of VLSI designs for the motion compensation block-matching algorithm," *Circuits and Systems, IEEE Transactions on*, vol. 36, pp. 1317-1325, 1989.
[5] S.-J. Kang, K.-R. Cho, and Y. H. Kim, "Motion compensated frame rate up-conversion using extended bilateral motion estimation," *Consumer Electronics, IEEE Transactions on*, vol. 53, pp. 1759-1767, 2007.
[6] B.-T. Choi, S.-H. Lee, and S.-J. Ko, "New frame rate up-conversion using bi-directional motion estimation," *Consumer Electronics, IEEE Transactions on*, vol. 46, pp. 603-609, 2000.
[7] S.-J. Kang, K.-R. Cho, and Y. H. Kim, "Motion compensated frame rate up-conversion using extended bilateral motion estimation," *Consumer Electronics, IEEE Transactions on*, vol. 53, pp. 1759-1767, 2007.
[8] B.-T. Choi, S.-H. Lee, and S.-J. Ko, "New frame rate up-conversion using bi-directional motion estimation," *Consumer Electronics, IEEE Transactions on*, vol. 46, pp. 603-609, 2000.
[9] D.-G. Yoo, S.-J. Kang, and Y. H. Kim, "Direction-select motion estimation for motion-compensated frame rate up-conversion," *Display Technology, Journal* of, vol. 9, pp. 840-850, 2013.
[10] S.-J. Kang, S. Yoo, and Y. H. Kim, "Dual motion estimation for frame rate up-conversion," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, pp. 1909-1914, 2010.
[11] W. Li and E. Salari, “Successive Elimination Algorithm for Motion Estimation,” Image process., *IEEE Transactions on*, vol .4, no. 1, pp. 105-107, 1995.
[12] C. K. Cheung and L. M. Po, “Normalized partial distortion algorithm for block motion estimation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 3, pp. 417-422, Apr. 2000.
[13] C. K. Cheung and L. M. Po, “Adjustable partial distortion search algorithm for fast block motion estimation,” *IEEE Trans. Circuits Syst. Video Technol*., vol. 13, no. 1, pp. 100–110, Jan. 2003.