HEVC Fast FME Algorithm using IME RD-Costs based Error Surface Fitting Scheme

Yunpeng Li [#], Zhenyu Liu [*1], Xiangyang Ji [†], Dongsheng Wang [*]

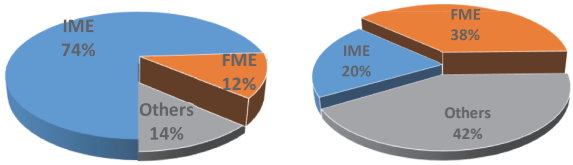
[#] *IMETU/* [†]*Department of Automation/* [*] *RIIT&TNList of Tsinghua University, China*
Lab 4-305, FIT Building, Tsinghua University, Beijing 100084, China
[1] liuzhenyu73@tsinghua.edu.cn

***Abstract*—Motion Estimation (ME), which is composed of integer motion estimation (IME) and fractional motion estimation (FME), is the most computational intensive module in HEVC encoding procedure. In this paper, a new fast fractional pixel motion search method, that is based on a six-parameter two-dimension error surface model, is proposed. In our proposal, by solving the over-determined equations, nine integer-pixel rate-distortion costs (RDC), including the best integer-pixel search candidate and its eight neighboring integer pixels, are used to estimate the six parameters in the model. Then, we can obtain the minimal position on the fitted error surface equation, which is the quarter-pixel accurate search center. We provided three kinds of search patterns in the quarter-pixel search stage, which could take a tradeoff between the computational complexity and the prediction accuracy. Experimental results demonstrate that, as compared with HM reference software (HM-15.0), the three proposed FME patterns could reduce 35.1%, 29.4%, and 22.5% encoding time, while the corresponding compression efficiency losses in terms of BDBR are 3.04%, 0.79%, and 0.43%, respectively.**

***Index Terms*—HEVC, FME, Error Surface Fitting, Fast Block Matching, Sub-accuracy Pixel**

I. Introduction

High Efficiency Video Coding (HEVC) [1] is the state-of-the-art video coding standard, developed by Joint Collaborative Team on Video Coding (JCT-VC). Motion Estimation (ME) algorithm plays the vital role in HEVC encoding procedure. That is, ME not only contributes most to the coding quality improvement, but also consumes more than 50% computational resource. In general, the ME processing is composed of two primary steps, i.e., integer-pixel motion estimation (IME) and fractional-pixel motion estimation (FME), both of which are based on the block matching algorithm (BMA). In specific, IME is first carried out to find the best integer-pixel accurate motion vector (MV) in the specified search window; next, around the best integer-pixel matching position, FME searches the sub-pixel candidates. For integer pixel search, the straightforward way is full search (FS) that calculates all integer vectors' costs within the limited search range and chooses the vector with the minimal one. Some fast search methods are also proposed, such as diamond search, square search, new three-step search [2] and so on, in which the first two have been accepted in HEVC. A widely-used fractional pixel search method is the hierarchical search scheme, which



(a) Full Search Mode (b) Fast Search Mode

Fig. 1. Run-time percentages of inter-slice encoding for full-search IME and fast-search IME configurations

firstly finds the best half-pixel precision vector by searching the nine neighboring half-pixel candidates centered on the best integer pixel, and then finding the best quarter-pixel vector in the same way.

In the procedure of ME, huge calculation incurs the serious cumbersome in the real-time encoder design. Figure 1 illustrates the complexity statistics of HM-15.0 encoding with two IME algorithms. We can observe that ME procedure occupies more than 58% of the whole encoding time. To reduce computational complexity, two strategies are employed: fast IME and fast FME. In fact, as shown by Fig. 1, after applying the fast IME algorithm, FME has became the primary computational intensive module in the encoder. While IME algorithm has gradually matured, it's important to develop fast FME algorithm. Complexity in FME comes from the interpolation for sub-pels generation and the RDC calculation. Previous experimental results [3] describe that blocks of real-world images are often smooth and gentle, which leads to a center-biased global minimum MV distribution. Du, et.al., [4] proposed the error surface model based fast FME. Then, several functions have been introduced to fit the error surface, and the best sub-pixel position was calculated by minimizing the functions. In [5], a 5-parameter error surface was put forward, of which parameters are obtained by five integer-pixel RDCs. Then, a refinement FME, which uses the minimal point of the 5-parameter error surface as the center, is processed. Dai, et.al., approximated the error surface by two 5-parameter models in half-pixel FME and quarter-pixel FME, respectively[6]. Hill, et.al., adopted a 6-parameter model estimated from 6 integer-pixel RDCs[7]. Dikbas, et.al., developed a 9-parameter cubic polynomial model to improve the precision[8]. In this paper, we propose a fast fractional pixel search method based on the error surface model. The contributions come from two aspects: First, the proposed quadratic polynomial error surface model is derived from the RDCs of the best integer

978-1-5090-5316-2/16/$31.00 © 2016 IEEE

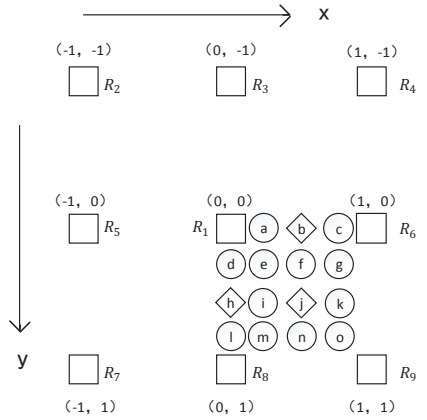


Fig. 2. Integer Pixels and Fractional Pixels (Square Dots : Integer Pixels, Diamond Dots : Half-Accuracy Pixels, Circle Dots : Quarter-Accuracy Pixels)

pixel and its eight integer neighbors. Second, we propose the simplified FME search patterns, which not only reduce the search position number, but also abridge the complexity of sub-pixel interpolation.

The rest of this paper is organized as follows: Section II elaborates the proposed fast fractional search algorithm. Section III provides the performance analysis of our fast FME, and further compares it with other previous works. Finally, Section IV gives the conclusions.

II. Quadratic Polynomial Error Surface Model Based FME Algorithm

In HEVC, ME is composed of integer pixel search and fractional pixel search, both of which return a best MV judged by RDC performance. In general, the RDC is calculated as follows,

$$RDC = SA(T)D(\vec{\Delta}) + \lambda \cdot R_{MV}(\vec{\Delta}), \tag{1}$$

in which, $\vec{\Delta} = (\Delta_x, \Delta_y)$ is the motion vector difference(MVD that is the difference between the motion vector and the motion vector predictor), λ is the Lagrange multiplier and R_{MV} represents the rate cost of MVD. SAD, denoting the sum of absolute difference between the original block and the predictive block, is used as the distortion cost in IME procedure. In FME, SATD is the matching distortion cost, which is the sum of absolute Hadamard Transform coefficients of prediction residues.

A. Error Surface Estimation with Nine Integer-Pixel RDCs

To reduce complexity of the existing FME algorithm, we model the 6-parameter error surface as

$$R(x, y) = Ax^2 + By^2 + Cxy + Dx + Ey + F, \tag{2}$$

where R represents the RDC of an integer MV, x and y denote the distance from the best integer MV in abscissa and ordinates, respectively. In our proposal, the nine integer pixel RDCs, including the best integer MV and its eight neighbors, are used to estimate the six parameters in (2), which is presented by the following matrix operation

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & -1 & -1 & 1 \\ 0 & 1 & 0 & 0 & -1 & 1 \\ 1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 0 & 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \\ E \\ F \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \\ R_7 \\ R_8 \\ R_9 \end{bmatrix}. \tag{3}$$

The RDCs of nine integer MVs, i.e., R_i ($i \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$) and their positions are shown as Fig. 2. For example, R_1 denotes the RDC of best integer MV. The corresponding x and y of R_1 are both 0s. R_i can be obtained in IME procedure.

For convenience, we rewrite the matrix operation (3) as the following form

$$\mathbf{Q}\vec{P} = \vec{R}. \tag{4}$$

For $\mathbf{Q}$ is a 9x6 matrix, obviously, (4) is an overdetermined equation. With the traditional convex optimization method, the solution of $\vec{P}$ is formulated as

$$\vec{P} = (\mathbf{Q}^T\mathbf{Q})^{-1}\mathbf{Q}^T\vec{R}, \tag{5}$$

where T denominates the transpose operation. $(\mathbf{Q}^T\mathbf{Q})^{-1}\mathbf{Q}^T$ is constant matrix. Actually, the parameter F in model (2) has no effect on the minimum position prediction, so we just drop this item. Finally, we have

$$\begin{bmatrix} A \\ B \\ B \\ D \\ E \end{bmatrix} = \begin{bmatrix} -1/3 & -1/3 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/4 & -1/6 & -1/6 \\ -1/3 & 1/6 & 0 & 0 & -1/6 \\ 1/6 & 1/6 & -1/4 & 1/6 & -1/6 \\ 1/6 & -1/3 & 0 & -1/6 & 0 \\ 1/6 & -1/3 & 0 & 1/6 & 0 \\ 1/6 & 1/6 & -1/4 & -1/6 & 1/6 \\ -1/3 & 1/6 & 0 & 0 & 1/6 \\ 1/6 & 1/6 & 1/4 & 1/6 & 1/6 \end{bmatrix}^T \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \\ R_7 \\ R_8 \\ R_9 \end{bmatrix} \tag{6}$$

Deriving the key parameters, we can deduce the minimum position in the error surface model by calculating the derivative of the function, i.e.,

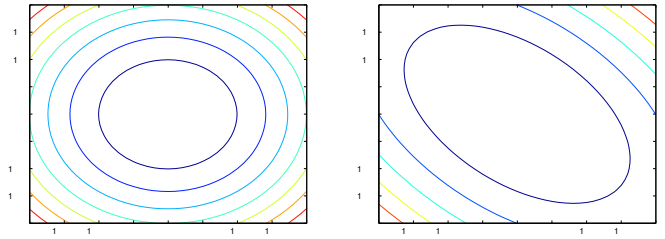
$$\frac{\partial R}{\partial x} = 0, \frac{\partial R}{\partial y} = 0. \tag{7}$$

In consequence, the minimum position is computed by

$$x_{\min} = \frac{2BD - CE}{C^2 - 4AB}, y_{min} = \frac{2AE - CD}{C^2 - 4AB}. \tag{8}$$

Because the quarter-pixel accuracy is enough, we can avoid divide operations in (8) by comparing the amplitude of numerator with the scaled amplitude of denominator.

Using the aforementioned minimum position as the search center, our FME refinement could make the tradeoff between the computational complexity and the compression efficiency. In specific, we proposed three search patterns: The first pattern merely searches the minimum position; The second pattern



(a) Error Surface Contour without Cross Term

(b) Error Surface Contour With Cross Term

Fig. 3. Contours of 5-Parameter and 6-Parameter Error Surface Models

searches the minimum position and its horizontal and vertical nearest four quarter-pixel positions; The third one searches the minimum position and all eight neighboring quarter-pixel positions. The performance of all these proposals will be presented in Section III.

B. Comparison of Different Methods

There are three models proposed for approximating the error surface, i.e., 5-parameter model, 6-parameter model and 9-parameter model. The 9-parameter model strives to improve the fitting accuracy by using the cubic terms. However, our experiments reveal that the cubic terms have little positive effect on search center prediction, while introducing more multiplications when calculating the minimum position. For the 5-parameter model, its computational complexity is the lowest in three competitors. However, without the cross term, i.e., $x \cdot y$, it can only fit the error surface like Fig. 3(a). In contrast, when the axles of elliptic contour are in the diagonal directions, as shown in Fig. 3(b), 6-parameter model is essential to fit the error surface.

The coding quality analysis of three typical models, including 5-parameter model, 6-parameter model using 6 integer-pixel RDCs, and our 6-parameter model using 9 integer-pixel RDCs, is illustrated in Table I. Twenty-five typical open test sequences, including Classes A-F, were tested with QP ={22, 27, 32, 37}. The metric of coding performance comparisons is BDPSNR, where the original HM-15.0 is adopted as the benchmark [9]. All models under test apply the 9-point FME search pattern, as described in Section II-A. Therefore, the time saving performance of all candidates matched each other. For the coding quality, it is nature that, because the 5-parameter model drops the term of $x \cdot y$, its coding quality is lower that other 6-parameter models. As compared with the 6-parameter model derived from 6 RDCs, our overdetermined system improves the performance by introducing more constrains in parameter estimation. Consequently, the coding picture quality is ameliorated by BDPSNR=+0.003dB averagely. This coding quality gap is dilated up to BDPSNR=+0.008dB in Class F test.

C. Interpolation Simplification

The original FME algorithm firstly searches the nine half-accuracy MVs centered on the best integer MV and calculates their RDCs to obtain the best half-accuracy MV. Then it chooses the best quarter-accuracy MV by calculating the nine

TABLE I
CODING PERFORMANCE COMPARISONS OF DIFFERENT ERROR SURFACE FITTING MODELS(UNIT:DB)

Algorithm	Class					
	A	B	C	D	E	F
M1	-0.0176	-0.0415	-0.1083	-0.1592	-0.0065	-0.1023
M2	-0.0142	-0.0059	-0.0176	-0.0218	-0.0069	-0.0722
Proposed	-0.0090	-0.0057	-0.0173	-0.0201	-0.0044	-0.0647

M1: 5-parameter model; M2: 6-parameter model using 6 integer-pixel RDCs; Proposed: 6-parameter model using 9 integer-pixel RDCs.

quarter-pixel MVs' RDCs centered on the best half-accuracy MV. As a matter of fact, many fractional pixel interpolations are carried out and the generated pixels should be stored for further distortion computation. Pixels classification is shown in fig. 2, where square dots represent integer pixels, diamond dots are half-pixels and circle dots stand for quarter-pixels. All fractional pixels belong to the integer pixel $(0, 0)$. For example, if the best integer pixel is $(0, 0)$ in fig. 2, we must interpolate the half pixels belonging to the eight search candidates around $(0, 0)$, i.e., "b" , "j" , "h" and so on. Then, if the best half-accuracy MV locates on the "j" in the picture, another eight candidates with quarter-pixel accuracy {"e", "f", "g", "k", "o", "n", "m", "i"} need to be interpolated and stored for quarter-accuracy pixel refinement.

In our proposal, the best quarter-accuracy MV could be estimated directly by finding the minimal position of error surface, which saves much interpolation computation and storage as a result. Let's use the same example as above i.e., the best integer MV is $(0, 0)$ and the minimum position of error surface locates on "j". If we adopt the first search pattern mentioned in Section II-A, only one half-pixel search candidate "j" should be calculated for the first step search. All the pixels to be interpolated of three proposed refinement patterns are {"j"}, {"j", "f", "k", "n", "i"} and {"j", "e", "f", "g", "k", "o", "n", "m", "i"}, respectively, that is much fewer than the original FME. Clearly, the proposed method simplifies the interpolation procedure.

III. EXPERIMENTS

The proposed method is integrated in HEVC reference test model HM-15.0. The test platform is Huawei RH5885, which combines Intel Xeon E7-4830-v2 2.20GHz processor and 128.0GB RAM. In total, we test twenty five video sequences from Class A to Class F with quantization parameters ranging {22,27,32,37}. The configuration of "encoder_lowdelay_main" is used and we choose fast search mode in integer pixel search method.

The quantitative coding efficiency analysis was conducted on the basis of the average PSNR (BDPSNR) gain and the average rate (BDBR) reduction[9]. The complexity reduction of our proposals is evaluated by the time saving performance of encoding process. Let T_{HM} denote the coding time consumed by HM-15.0 and T_{FM} be the time used by our error surface fitting based fast FME algorithm, and ΔT represents encoding time reduction, which is written as $\Delta T = \frac{T_{HM} - T_{FM}}{T_{HM}} \times 100\%$.

The performance comparisons are presented in Table II. In Table II, we present the performance statistics of three pro-

TABLE II
CODING PERFORMANCE OF DIFFERENT METHODS(BDPSNR UNIT: DB; BDBR UNIT: %; ΔT UNIT: %)

Class	Sequence	Existing[6]			Proposed I			Proposed II			Proposed III		
		BDPSNR	BDBR	ΔT	BDPSNR	BDBR	ΔT	BDPSNR	BDBR	ΔT	BDPSNR	BDBR	ΔT
A	PeopleOnStreet	-0.1875	4.25	29.7	-0.1398	3.47	29.9	-0.0358	0.90	24.6	-0.0173	0.43	19.8
	Traffic	-0.1736	5.39	37.2	-0.1302	4.70	37.9	-0.0069	0.23	32.0	-0.0008	0.03	24.2
B	BasketballDrive	-0.0501	1.70	31.8	-0.0176	0.95	32.1	-0.0143	0.63	26.8	-0.0111	0.45	21.0
	BQTerrace	-0.0382	1.42	27.6	-0.0100	0.62	27.0	-0.0075	0.47	22.2	-0.0032	0.21	18.6
	Cactus	-0.0407	1.43	30.9	-0.0153	0.61	31.3	-0.0093	0.44	25.6	-0.0029	0.16	21.5
	Kimono	-0.0245	1.13	30.4	-0.0090	0.37	31.2	-0.0091	0.31	24.5	-0.0051	0.17	18.9
	ParkScene	-0.0413	1.34	34.0	-0.0156	0.61	33.8	-0.0115	0.40	28.8	-0.0051	0.18	22.5
	Tennis	-0.0586	1.60	32.3	-0.0304	0.72	33.8	-0.0168	0.61	23.4	-0.0093	0.34	17.2
C	BasketballDrill	-0.1932	4.54	31.0	-0.1199	3.74	29.6	-0.0286	0.80	25.1	-0.0107	0.30	18.6
	BasketballDrillText	-0.1929	4.71	26.9	-0.1546	3.99	27.0	-0.0473	1.21	21.7	-0.0293	0.75	16.1
	BQMall	-0.1630	4.15	27.5	-0.1267	3.37	27.6	-0.0299	0.80	21.6	-0.0136	0.36	16.4
	PartyScene	-0.2946	6.48	28.7	-0.2205	5.79	29.5	-0.0364	0.91	22.8	-0.0155	0.38	16.9
	RaceHorsesC	-0.1899	5.01	27.1	-0.1489	4.30	26.9	-0.0354	0.96	19.9	-0.0199	0.54	15.4
D	BasketballPass	-0.2135	4.77	35.6	-0.1697	4.00	36.0	-0.0460	1.10	30.2	-0.0289	0.69	23.7
	BlowingBubbles	-0.1916	5.44	37.0	-0.1586	4.59	37.4	-0.0302	0.86	28.4	-0.0163	0.45	22.6
	BQSquare	-0.4994	6.43	33.8	-0.4002	5.68	34.1	-0.0503	1.41	28.9	-0.0191	0.52	23.1
	RaceHorses	-0.2971	6.50	27.4	-0.2401	5.74	27.4	-0.0576	1.32	22.8	-0.0327	0.77	17.0
	Keiba	-0.2903	6.05	30.5	-0.2213	5.26	29.9	-0.0523	1.12	26.2	-0.0266	0.57	20.1
E	Vidyo1	-0.1609	3.74	44.3	-0.0819	2.93	44.5	-0.0091	0.35	40.5	-0.0034	0.19	31.7
	Vidyo3	-0.1736	4.91	46.1	-0.1206	4.17	46.4	-0.0061	0.27	40.8	-0.0005	0.03	30.5
	Vidyo4	-0.1347	4.03	41.6	-0.0807	3.24	41.5	-0.0122	0.48	38.0	-0.0073	0.39	27.8
	Johnny	-0.1089	4.16	43.6	-0.0749	3.33	45.6	-0.0051	0.31	39.0	-0.0028	0.09	28.1
	KristenAndSara	-0.1102	3.38	43.8	-0.0702	2.59	42.1	-0.0159	0.63	40.5	-0.0080	0.31	29.4
F	SlideEditing	-0.0915	1.34	60.7	-0.0730	0.51	59.7	-0.1383	1.00	51.8	-0.1111	0.80	38.2
	ChinaSpeed	-0.0445	1.28	33.9	-0.0317	0.63	34.2	-0.1165	2.30	28.0	-0.0783	1.54	22.2
	Average	-0.1586	3.81	34.9	-0.1145	3.04	35.1	-0.0331	0.79	29.4	-0.0192	0.43	22.5

Proposed I: one-pixel refinement; Proposed II: 5-pixel refinement; Proposed III: 9-pixel refinement

posed search patterns. In specific, let (mv_x, mv_y) represents the quater-pixel position closest to (8). "Proposed I" just calculates the RDC in (mv_x, mv_y). "Proposed II" searches (mv_x, mv_y) and its nearest neighbors $(mv_x \pm 1/4, mv_y)$ and $(mv_x, mv_y \pm 1/4)$. "Proposed III" searches (mv_x, mv_y) and all 8 neighbors. We can see that, "Proposed I" achieved the most computation saving (35.1%), while its rate increase is also the biggest (BDBR=+3.04%). In contrast, "Proposed III" has the best coding quality (BDBR=+0.43%) with 22.5% encoding time saving. The performance of "Proposed II" is in the middle of "Proposed I" and "Proposed III". We implement the work in [6], which proposed a two-round 5-parameter error surface model. In the first round estimation, it carries out a half-pixel FME to derive the RDC of the half-pixel minimum position. In the second round processing, the minimum position is refined in quarter-pixel precision based on the new half-pixel RDC. As expected, its time saving performance is approaching to our "Proposed I". However, because our method adopts 6-parameter model and introduces more constraints in model estimation, the coding quality of "Proposed I" is superiors over the counterpart [6] with BDPSNR=+0.044dB in average.

IV. CONCLUSION

This paper presents a fast fractional pixel search algorithm based on the 6-parameter error surface fitting model. In our proposal, nine integer-pixel RDCs, including the best integer-pixel search candidate and its eight neighboring pixels, are used to estimate the key parameters in the model by solving the over-determined equations. Then, using the minimal position of error surface as the quarter-pixel search center, we further provide three patterns to make the tradeoff between the computation intensity and the compression efficiency. The low complexity configuration achieved 35.1% time saving at the cost of BDBR=+3.04% rate increasement. In contrast, the high complexity mode has the minimum coding loss at BDBR=+0.43%, while the time saving is merely 22.5%. The moderate configuration balances these two metrics, with BDBR=+0.79% and $\Delta T = 29.4\%$.

REFERENCES

[1] G. J. Sullivan, J. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[2] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438–442, 1994.

[3] L. M. Po and W. C. Ma, "A new center-biased search algorithm for block motion estimation," in *International Conference on Image Processing, 1995. Proceedings*, 1995, pp. 410–410.

[4] C. Du, Y. He, and J. Zheng, "Pphps: a parabolic prediction-based, fast half-pixel search algorithm for very low bit-rate moving-picture coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 6, pp. 514–518, 2003.

[5] J. F. Chang and J. J. Leou, "A quadratic prediction based fractional-pixel motion estimation algorithm for h.264 ," in *IEEE International Symposium on Multimedia*, 2005, pp. 1074–1089.

[6] W. Dai, O. C. Au, C. Pang, and L. Sun, "A novel fast two step sub-pixel motion estimation algorithm in hevc," vol. 22, no. 10, pp. 1197–1200, 2012.

[7] P. R. Hill, T. K. Chiew, D. R. Bull, and C. N. Canagarajah, "Interpolation free subpixel accuracy motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 12, pp. 1519–1526, 2007.

[8] S. Dikbas, T. Arici, and Y. Altunbasak, "Fast motion estimation with interpolation-free sub-sample accuracy," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 7, pp. 1047–1051, 2010.

[9] G. Bjontegaard, "Calculation of average psnr difference between rd-curves," 2001.