Vision, Video and Graphics (2005)
E. Trucco, M. Chantler (Editors)

Linear Hashtable Method and Predicted Hexagonal Search Algorithm with Moments Invariant

Yunsong Wu[1], Graham Megson[1], Zhengang Nie[3], Xuan Liu[2]

[1]School of Systems Engineering, Reading University, Reading, UK
[2]Department of Computer Science, Loughborough University, UK
[3]Beihang University, China

Abstract

this paper presents a novel Linear Hashtable Method Predicted Hexagonal Search (LHMPHS) method for block base motion compensation on the basis of research from previous algorithm. Hashtable is used in video compression. Motion vectors produced by Linear Hashtable Motion Estimation Algorithm (LHMEA) are used as predictors for HEXBS. Moments invariants are also tested in hashtable to prove the more information moments have, the better it is. Experimental results show that the proposed algorithm can offer the same compression rate as the Full Search and fastest than all investigated algorithms, while the PSNR is high. LHMPHS has significant improvement on HEXBS and shows a direction for improving other fast motion estimation algorithms.

Categories and Subject Descriptors (according to ACM CCS): I.4.2 [Image Processing and Computer Vision]: Compression (Coding)

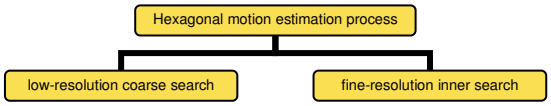
1 Introduction

In this paper, we propose a Linear Hashtable Motion Estimation Algorithm (LHMEA) to predict motion vectors for inter-coding. The objective of our motion estimation scheme is to achieve good quality video with very low computational complexity. Our method attempts to predict the motion vectors using linear algorithm. It uses hashtable method into video compression. After investigating of most traditional and on the edge motion estimation methods, I used latest optimization criterion and prediction search method. Spatially MBs' information is used to generate best motion vectors. I also combine the LHMEA with Hexagonal Search by motion predictor method. Hexagonal Search is one of best motion estimation method currently. The new method improved by me achieves best results so far. The main contributions of this paper are (1) it uses hashtable concept into video compression which uses several variables to represent whole MB [GF01]. This shows a direction for future research. (2) Linear Algorithm is used in video compression. This will improve speed; also leave space for parallel coding. (3) LHMEA is combined with hexagonal method. A new LHMEA predicted hexagonal method is proposed, which make up for drawback of coarse search of hexagonal search. (4) Spatially related MB's information is used not only in coarse search but also inner fine search. There are a large number of motion prediction algorithms. We only focus on one class of such algorithms, called the Block Matching Algorithms, which is widely used in MPEG2, MPEG4, and H.263. By partitioning a current frame into non-overlapping macroblocks with equal size, block-matching method attempts to find a block from a reference frame (past or future frame) that best matches a predefined block in the current frame. Matching is performed by moving and comparing with a criterion, which is called the MAE mean absolute error/difference in our research. The MB (macroblock) in the reference frame moves inside a search window centered on the position of the current block in the current frame. The best matched block producing the minimum distortion is found within the search window in the reference frame. However, the motion estimation is quite computationally intensive and can consume up to 80% of the computational power of the encoder if the full search is used. It is highly desired to significantly speed up the process without sacrificing the distortion seriously. Many computationally efficient variants were developed, among which are typically Two

Level Search (TS), Two Dimensional Logarithmic Search (DLS) and Subsample Search (SS) [ZE00], the Three-Step search (TSS), Four-Step Search (4SS) [LW96], Block-Based Gradient Descent Search (BBGDS) [LE96], and Diamond Search (DS) [SK00], [JSM*98] algorithms. A very interesting method called HEXBS has been proposed by Ce Zhu, Xiao Lin, and Lap-Pui Chau [CXL02]**.** There are some variant HEXBS, such as Enhanced Hexagonal method[CXL03] and Hexagonal method with Fast Inner Search[CXL*04].

1.1 Hexagonal Algorithm

Hexagonal method is an improved method based on DS (Diamond Search). It has shown the significant improvement over other fast algorithms such as DS. In contrast with the DS that uses a diamond search pattern, the HEXBS adopts a hexagonal search pattern to achieve faster processing due to fewer search points being evaluated. The motion estimation process normally comprises two steps. The low-resolution coarse search to identify a small area where the best motion vector is expected to lie, and then followed by fine-resolution inner search to select the best motion vector in the located small region.



Most fast algorithms focus on speeding up the coarse search by taking various smart ways to reduce the number of search points in identifying a small area for inner search. There are two main directions to improve the coarse search,

Coarse search improvement

1. usage of predictors [CXL*04], [PLG*04]
2. early termination [PLG*04]

In [CXL*04] a new algorithm was introduced on Hexagonal search, which is similar as Motion Vector Field Adaptive Search Technique (MVFAST) [AOM01] based on DS. The algorithm has significantly improved the preexisting Hexagonal Search algorithm both in image quality and speed up by initially considering a small set of predictors, namely the (0,0) motion vector and the motion vectors of the three spatially adjacent blocks(left, top, top-right) as possible motion vector predictor candidates. Modified Hexagonal pattern used the best motion vector predictor candidate as the center of search. In [[AOM02], [HA02]] it was proposed a prediction set. In general, we can state that the blocks correlated with the current one, which are likely to undergo the same motion, can be divided into three categories as in Figure.1.

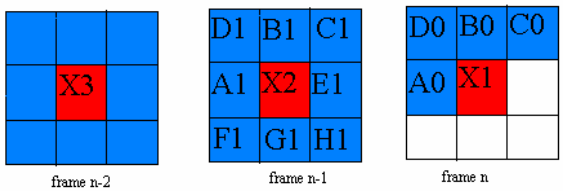


Figure. 1. *Blocks correlated with the current one*

(1) Spatially correlated blocks (A0, B0, C0, D0),
(2) Neighboring blocks in the previous frame (A1, B1, C1, D1, E1, F1, G1, H1)
(3) Co-located blocks in the previous two frames (X2 and X3), which provide the Acceleration MV.
Except for coarse search improvement, Inner search improvement includes:

1. 4 points [CXL*04]
2. 8 points [PLG*04]
3. Inner group search [PLG*04]

2 Linear Hashtable Method Predicted Hexagonal Search (LHMPHS)

Most of current Hexagonal search algorithms predictive methods focus on relations between current frame and previous frames. What we want to do is to find a fast method which discovers the predictor from current frame information. It uses spatially related MB or pixels' information. It is fast, accurate and independent on finding right predictors.
So we designed a vector hashtable lookup algorithm matching algorithm which is a more efficient method to perform an exhaustive search: it considers every macroblock in the search window. This block-matching algorithm calculates each block to set up a hashtable. It is a dictionary in which keys are mapped to array positions by a hash function. It is a method of compressing image data comprising the steps of generating a set of motion vectors representative of one or more image frames, generating, by means of a predetermined hash function a set of hash values corresponding to said motion vectors, and storing as a codebook said hash values in the form of a table. We try to find as few as possible variables to represent the whole macroblock. The details of setting up hashtable are as following:

For the purpose of the present description, assume that the size of the hash table is fixed (i.e. 'static hashing' as opposed to 'dynamic hashing' in which the table size may vary). The address of a data item x stored within the hash table may be computed by evaluating the hash function H(x). Typically, hash tables are partitioned into b 'buckets', with each bucket consisting of s 'slots' Each slot is capable of storing exactly one data item, and in LHMEA that s=2, i.e. each bucket stores just one data item.

The construction of the hash function H(o) is the most crucial aspect of designing a hash table. Not only

should H(o) be easy to compute, but it should also ideally generate a unique address within the hashtable for each argument. It is not possible for the hash table to hold every possible value of the argument. Hence, it has been found that collisions often occur. i. e. H(x)=H(y) for two data items $\{x, y | x \neq y\}$. Another problem is that of overflow, whereby a data item is mapped by H(o) into an existing bucket that is already full. Ideally, therefore, hash functions should be designed to minimize the possibility of both collisions and overflow. It has been found that there are advantages to encoding groups of image sequences, as opposed to encoding individual samples. A technique known as vector quantization (VQ) utilizes this finding and offers a way of performing lossy compression along the way.

VQ is essentially the multi-dimensional generalization of scalar quantization, as is commonly employed in analog-to-digital conversion processes. In analytical terms, if X is an N-dimensional source vector, then VQ is a mapping M such that:

$$M : H^N -> L$$

where L is an L-dimensional set, L<N, such that $L = \{Y_1, ..., Y_N\}$, and the $L_i \in H^N$ for i=1,…,L. L is usually termed the 'codebook', and the Y_i the 'code vectors'. The VQ operator Q partitions H^N into L disjoint and exhaustive regions $\{P_1, ...P_L\}$, each of which has a single coarse-grained representation.

From an implementation perspective, encoding involves a two-stage process:

1. Hashtable generation, in which a decision is made on the number of entries in the hashtable. Representative code vectors from the I-frame are computed (using linear hashtable training algorithms) and stored in hashtable. Clearly, the larger the hashtable, the less quantization error during encoding and look-up. It follows, therefore, that the minimum bound on the size of hashtable should be at least equal to the maximum number of motion vectors that any subsequent predicted frame will require. Thus video fidelity becomes a function of the size of the hashtable.

2, Hash Table loading, in which the Vector Quantized Hashing Table(VQHT) bucket slots are filled up by feeding every possible source vector (macroblock) from the I-frame through the hash function and storing it (together with its co-ordinates) in its appropriate bucket. Bucket slots are filled up sequentially in this manner. Through some preprocessing steps, "integral projections" are calculated for each macroblock. These projections are different according to different algorithm. The aim of these algorithms is to find best projection function. For example, in my report, I represent 2 algorithms. Each has 2 projections, one of them is the massive projection, which is a scalar denoting the sum of all pixels in the macroblock. It is also DC coefficient of macroblock. Another is a of Y=ax+b (y is luminance, x is the location.) Each of these projections is mathematically related to the error metric. Under certain conditions, the value of the projection indicates whether the candidate macroblock will do better than the best-so-far match. The major algorithm we discuss here is linear algorithm.

2.1 Linear Hashtable Motion Estimation Algorithm (LHMEA)

Linear Algorithm is most beautiful, easy and fast to calculate on computer because the construction of computer calculator based on additions. So if most of calculations of video compression are done on linear algorithm, we can save lots of time on compression. It is also very easy to put on parallel machines in the future, which will benefit real time encoding. In the program, we try to use polynomial approximation to get such result y=mx+c; y is luminance value of all pixels, x is the location of pixel in macroblocks. The way of scan y is from left to right, from top to button. Coefficients m and c are what we are looking for.

$$m = \frac{N * \sum_{i=0}^{N} (x_i * y_i) - \sum_{i=0}^{N} x_i * \sum_{i=0}^{N} y_i}{N * \sum_{i=0}^{N} x_i^{\,2} - \sum_{i=0}^{N} x_i * \sum_{i=0}^{N} x_i}$$

$$c = \frac{\sum_{i=0}^{N} y_i * \sum_{i=0}^{N} x_i^{\,2} - \sum_{i=0}^{N} x_i * \sum_{i=0}^{N} x_i * y_i}{N * \sum_{i=0}^{N} x_i^{\,2} - \sum_{i=0}^{N} x_i * \sum_{i=0}^{N} x_i}$$

in this way, we initially realized the way to calculate the hashtable. In previous methods, when we try to find a block that best matches a predefined block in the current frame, matching was performed by SAD (calculating difference between current block and reference block). In Linear Hashtable Motion Estimation Algorithm (LHMEA), we only need compare two coefficients of two blocks. In current existing methods, the MB moves inside a search window centered around the position of the current block in the current frame. In LHMEA, the coefficients move inside hashtable to find matched blocks. If coefficients are powerful enough to hold enough information of MB, motion estimators should be accurate. So LHMEA increases lots of speed, accuracy and will make a new era of video encoding.

2.2 Linear Hashtable Method Predicted Hexagonal Search

After motion estimators are generated by LHMEA, they will be used as predictors for hexagonal algorithm for coarsely search. These predictors are

different from all previous predictors. They are based on full search and current frame only. Because LHMEA is linear algorithm, it is fast. Because the predictors generated are accurate, it will improve Hexagonal method without too much delay in speed.

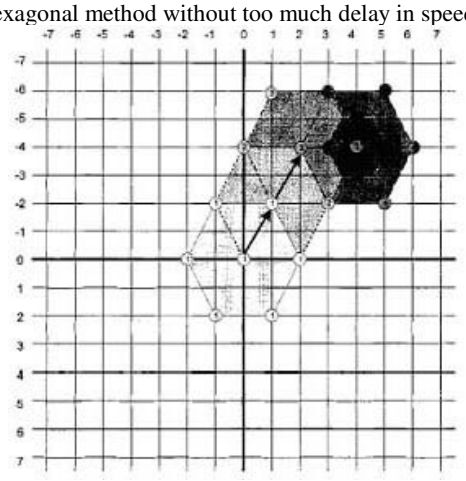


Figure.2.*Original Hexagonal Coarse Search*

The original Hexagonal Search moved step by step, maximum two pixels per step, but in our proposed method, The LHMEA motion vectors are used to move hexagon directly to the area where near to the pixel whose MB distortion is smallest.
This saved lots of computation on low-resolution coarse search.

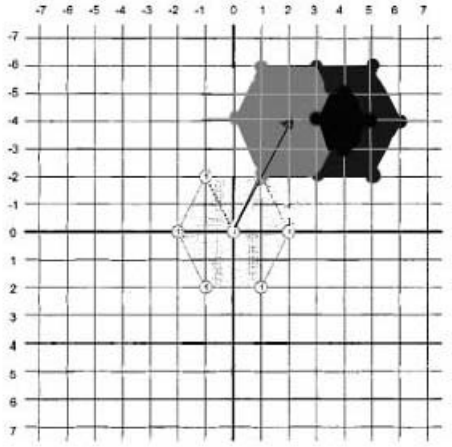


Figure.3 *Proposed Hexagonal Coarse Search*

In the Figure below, we compared Full Search (FS), Linear Hashtable Motion Estimation Algorithm (LHMEA), Subsample Search (SS), Two Level Search (TLS), Logarithmic Search (LS) and three kinds of Hexagonal Search Algorithms. Three Hexagonal Search Algorithms are Hexagonal Search(HS), Linear Hashtable Method Predicted Hexagonal Search (LHMPHS) and Hexagonal Search With Median Predictor of Spatially Adjacent Blocks(left, up and upright blocks what are respectively named A0, B0 and C0 in Figure. 1. (HSM) [AOM01]. All HS algorithms used 6-side-based fast inner search [PLG*04] and early termination criteria [AOM01] mentioned in our paper. All the data here refer to P frames only. The HS can achieve nearly the same PSNR as FS and only takes 10% time of FS. The LHMPHS is better than HS without predictor on compression rate when time and PSNR are the same. HSM is the best algorithm. But if we can find better coefficients in the hashtable to represent MB, the hashtable will have a wonderful future.

We also tried flower garden data stream. It is under the same condition as Table Tennis. As in the figure below, we can see LHMPHS is better than HS, while worse than HSM on both compression rate and PSNR of P frames.

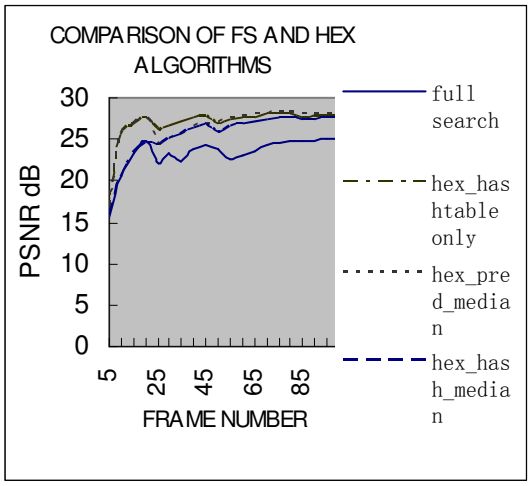


Figure. 4. *Comparison of PSNR between FS (full search), HS (hex_hashtable only), LHMPHS (hex_pred_median), and HSM (hex_hash_median) (based on 5-100 frames of Flower Garden)*

But in the following figure about PSNR from 5 to 100 frames of Table Tennis data stream, all the data algorithms' PSNR are the same. It means LHMEA works better on large motion vector video stream.

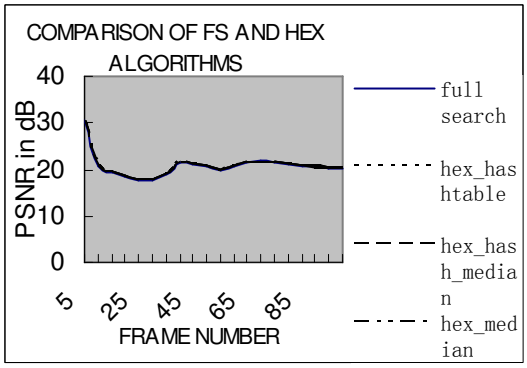
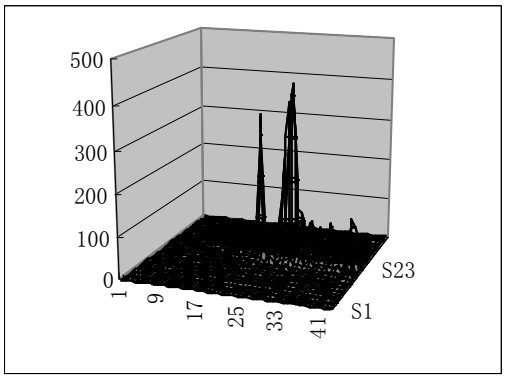
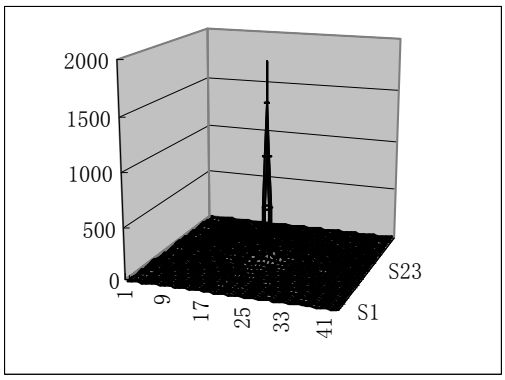


Figure. 5. *Comparison of PSNR between FS(full search), HS(hex_hashtable only), HMPHS(hex_pred_median), HSM(hex_hash_median) (based on 5-100 frames of Table Tennis)*

The FS, HS, LHMPHS, HSM are certain center biased algorithms. This is also basis of several other algorithms. It was based on the fact that for most sequences motion vectors were concentrated in a small area around the center of the search. This can also be seen in the figures below. Unfortunately for some sequences this is not always true as can be seen in the flower garden figure 6.(a), which implies that these algorithms will have reduced performance in such cases. From additional simulations we can observe that predictor seems to have a much higher correlation with the current motion vector than (0,0) even for non-center biased sequences such as the Flower Garden mentioned previously. This suggests that, instead of initially examining the (0,0) position, we could achieve better results if the LHMEA predictor is examined first and given higher priority with the use of early termination threshold.



(a)



(b)

Figure.6. *Motion vector distribution in (a) flower garden and (b) table tennis using hexagonal algorithm with LHMEA predictor.*

2.2 Moments Invariants.

Except for the coefficients from the Linear Algorithm, we put moments invariant into the hashtable as a test. The set of moments we are considering is invariant to translation, rotation, and scale change. We consider moments represent a lot more information than the coefficients m and c that we proposed in LHMEA. As we can see from the experimental result, moments have some improvement on hashtable method.

where $\bar{x} = \frac{m_{10}}{m_{00}}$ and $\bar{y} = \frac{m_{01}}{m_{00}}$

The normalized central moments, denoted η_{pq}, are defined as

$$\eta_{pq} = \frac{\mu_{pq}}{\mu^{\gamma}_{00}} \quad \text{where} \quad \gamma = \frac{p+q}{2} + 1 \quad \text{for}$$

p+q=2,3,….

A set of seven invariant moments can be derived from the second and third moments.

$$\phi_1 = \eta_{20} + \eta_{02}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}) * \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) * \left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]$$

$$\phi_6 = (\eta_{20} - \eta_{02})\left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12}) * \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03}) * \left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]$$

Table I shows experiment result using three different algorithms: LAHSBTPA without moments; LAHSBTPA with 2 moments ϕ_1 and ϕ_2 in hashtable; and LAHSBTPA with 7 moments in hashtable. The experiments result proves that invariant moments used in hashtable help increase compression rate and PSNR on the cost of compression time. It means if we can find better coefficients in hashtable, the experimental result can be further improved.

TABLE I
Comparison of compression rate, time and PSNR among TPA with different number of moments in hashtable (based on 150 frames of Table Tennis)

Data Steam	Table Tennis	Table Tennis	Table Tennis
Moment Test	No moments	With 2 Moments	With 7 Moments
compression time(s)	16	32	43
compression rate	94	94	95
average P frame PSNR	40.2	40.6	40.6

3 Summary

In the paper we proposed a new algorithm called Linear Hashtable Motion Estimation Algorithm (LHMEA) in video compression. It uses linear algorithm to set up hashtable. The algorithm searches in hashtable to find motion estimator instead of by FS. Then the motion estimator it generated will be sent to Hexagonal algorithm, which is the best motion estimation algorithm, as predictor. In this way, it is improved both in quality and speed of motion estimation. Moments invariants are used to prove the more information hashtable has, the better it is. The key point in the method is to find suitable coefficients to represent whole MB. The more information the coefficients in hashtable hold about pictures, the better result LHMPHS will get. This also leaves space for future development.

References

[AOM01] Alexis M. Tourapis, Oscar C. Au, Ming L. Liou: Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) Enhancing Block Based Motion Estimation. proceedings of Visual Communications and Image Processing, San Jose, CA, January (2001)

[AOM02] A. M. Tourapis, O. C. Au and M. L. Liou: Highly Efficient Predictive Zonal Algorithms for Fast Block. Matching Motion Estimation. IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, No.10, pp 934-947, (October 2002)

[CXL02]. Ce Zhu, Xiao Lin, and Lap-Pui Chau: Hexagon-Based Search Pattern for Fast Block Motion Estimation. IEEE Trans on circuits and systems for video technology, Vol. 12, No. 5, (May 2002)

[CXL03]. C. Zhu, X. Lin and L.P. Chau: An Enhanced Hexagonal Search Algorithm for Block Motion Estimation. IEEE International Symposium on Circuits and Systems, ISCAS2003, Bangkok, Thailand, (May 2003)

[CXL*04]. Ce Zhu, Xiao Lin, Lappui Chau, and Lai-Man Po: Enhanced Hexagonal Search for Fast Block Motion Estimation. IEEE Trans on circuits and systems for video technology, Vol. 14, No. 10, (Oct 2004)

[GF01] Graham Megson & F.N.Alavi Patent 0111627.6 -- for SALGEN Systems Ltd (2001)

[HA02] H-Y C. Tourapis, A. M. Tourapis: Fast Motion Estimation within the JVT codec. Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG 5th meeting, Geneva, Switzerland. (09-17 October 2002)

[JSM*98]. J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim: A novel unrestricted center-biased diamond search algorithm for block motion estimation. IEEE Trans. Circuits and systems for video technology, vol. 8, pp. 369–377, (Aug. 1998)

[LE96]. L. K. Liu and E. Feig: A block-based gradient descent search algorithm for block motion estimation in video coding," IEEE Trans. Circuits Syst. Video Technol., vol. 6, pp. 419–423, (Aug. 1996.)

[LW96]. L. M. Po and W. C. Ma: A novel four-step search algorithm for fast block motion estimation," IEEE and systems for video technology, vol. 6, pp. 313–317, (June 1996.)

[PLG*04] Paolo De Pascalis, Luca Pezzoni, Gian Antonio Mian and Daniele Bagni: Fast Motion Estimation With Size-Based Predictors Selection Hexagon Search In H.264/AVC encoding. EUSIPCO (2004)

[RA02] Rafael C. Gonzalez: Digital Image Processing second edition. Prentice Hall (2002)

[SK00]. S. Zhu and K.-K. Ma: A new diamond search algorithm for fast blockmatching motion estimation. IEEE Trans. Image Processing, vol. 9, pp. 287–290, (Feb. 2000.)

[ZE00]. Ze-Nian li Lecture Note of Computer Vision on personal website (2000)

Appendix:

	EXHAUSTIVE	SUBSAMPLE	TWOLEVEL	LOGARITHMIC	Vector Hashtable	Hex No Predictor	Pred_Hashtable	Pred_Median
						Inner Group	Inner Group	Inner Group
						Hex_near	Hex_near	Hex_near
Compression Time(s)	11	4	3	1	3	1	1	1
(fps)	2.3726	6.4286	7.3171	24.7706	8.4112	19.4245	14.2857	17.5325
Compression Rate	48	48	48	42	24	37	39	42
PSNR	21.3	21.3	21.3	21.8	24.9	21.2	21.2	21.2