Clustering Based Search Algorithm For Motion Estimation

Ke Chen, Zhong Zhou, Wei Wu
State Key Laboratory of Virtual Reality Technology and Systems
School of Computer Science and Engineering, Beihang University
Beijing, China
e-mail: {chenke, zz, wuwei}@vrlab.buaa.edu.cn

***Abstract*—Motion estimation is the key part of video compression since it removes the temporal redundancy within frames and significantly affects the encoding quality and efficiency. In this paper, a novel fast motion estimation algorithm named Clustering Based Search algorithm is proposed, which is the first to define the clustering feature of motion vectors in a sequence. The proposed algorithm periodically counts the motion vectors of past blocks to make progressive clustering statistics, and then utilizes the clusters as motion vector predictors for the following blocks. It is found to be much more efficient for one block to find the best-matched candidate with the predictors. Compared with the mainstream search algorithms, this algorithm is almost the most efficient one, 35 times faster in average than the full search algorithm, while its mean-square error is even competitively close to that of the full search algorithm.**

Keywords-motion estimation; clustering; search algorithm; video compression

I. INTRODUCTION

In most existing international video standards, such as the ISO MPEG series and the ITU-T H.26X series, motion estimation (ME) has been adopted to remove temporal redundancy within frames and thus to provide coding systems with high compression ratio. The popular ME fashion is trying to find the alike area in the reference frame corresponding to the one in the current frame based on blocks and is usually called block matching motion estimation (BMME) in video codecs.

Among numerous BMME algorithms, full search (FS), which compares all the candidates in the search window and finds the best-matched block, is the one with the minimum error but usually with the highest computation due to the thorough candidate matching. In order to reduce the computation, varieties of fast motion estimation algorithms have been proposed, such as three step search (TSS) [1], new three step search (NTSS) [2], four step search (FSS) [3], diamond search (DS) [4], hexagon search (HS) [5] and some other algorithms. These fast algorithms apply different search patterns and search strategies to reduce the search candidates, but their accuracy inevitably degrades, especially in video sequences with high motion.

Besides search patterns, fast motion estimation algorithms always utilize motion vector predictors to narrow the search range and thus to reduce the computation. According to the spatial correlation of block motion, the motion vector of one block is usually correlated to those of its neighboring blocks. Inspired by this, some predictive motion estimation algorithms have been put forward, such as predictive line search (PLS) [6], hybrid unsymmetrical cross multi-hexagon grid search (UMHexagonS) [7], predictive intensive direction searching (PIDS) [8], simulated annealing adaptive search (SAAS) [9] and other predictive algorithms. Predictive motion estimation algorithms use the motion vectors of the spatial/temporal neighboring blocks to build an initial predictor for the motion vector of the current block. These algorithms can narrow the search scope as well as the computation, but the search accuracy and the speed always ask for a tradeoff.

Even more than the facts above that have been found, in fact, those blocks that are not neighbors may also have a similar motion due to the similar depth or object segments of the scene. Consequently, a frame of a normal video sequence usually only has several clusters of motion vectors since normally the adjacent frames do not involve many arbitrary big movements in the scene. The main contribution of this paper is to periodically make clustering statistics on past motion vectors and to provide efficient predictors with most possibilities for the motion estimation of the following blocks. It's usually probable for one block to find the best-matched candidate quickly with the predictors. To the best of our knowledge, few works have been done before on the clustering statistics of motion vectors for predictive motion estimation.

The rest of this paper is organized as follows. In Section 2, our motivation and the clustering feature of motion vectors are described. Section 3 gives the clustering definition of motion vectors. Section 4 introduces the proposed Clustering Based Search algorithm. The experiment results are illustrated in Section 5. Finally, Section 6 draws a conclusion.

II. MOTIVATION

Traditional motion estimation algorithms usually exploit the motion vector similarity in neighboring blocks. However, those blocks that are not neighbors may also have a similar motion due to the depth or object segments of the scene. Generally speaking, the motion vectors of blocks in the same segment will be correlated and the motion vectors of blocks at the same depth will also be correlated. The motion vectors of a frame in a normal video sequence will have several clusters, since adjacent frames will not involve many big arbitrary movements in the scene.

978-0-7695-4711-4/12 $26.00 © 2012 IEEE
DOI 10.1109/ICME.2012.88

We count the motion vectors between two adjacent frames in the video sequence Aspen [10]. Fig. 1 illustrates the histogram of motion vectors computed by FS algorithm with search range (-8, 7) and block size 16×16. Each point in plane *xoy* denotes a motion vector (*x*, *y*) and the count axis shows the number of the blocks whose motion vector is (*x*, *y*). From Fig. 1, we can observe that the motion vectors follow a two-dimensional discrete distribution, and there exist several peaks i.e. the counts of some vectors and their close neighboring vectors are much higher than others.

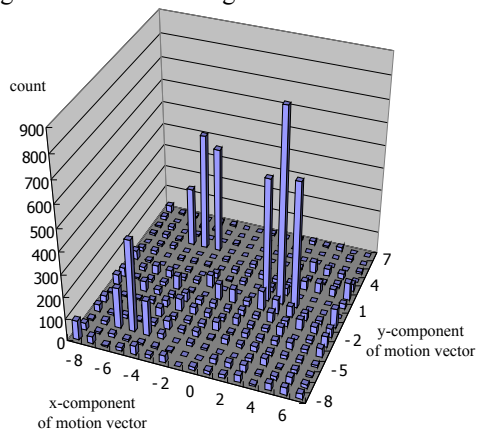

Figure 1. The histogram of motion vectors of adjacent frames in Aspen sequence [10].

To investigate the distance between those motion vectors with high counts, we choose the Manhattan distance as the metric. We select motion vectors (-6, -5), (0, 3) and (4, -4) at the three peaks in Fig. 1, and regard them as the representative vectors of G_1, G_2 and G_3 respectively. Then, the *k*-means algorithm is performed to classify the other motion vectors with count more than 150 (the average value of the motion vectors' counts) into the three groups G_1, G_2 and G_3. Consequently, the three groups are G_1={(-6, -4), (-6, -5), (-6, -6)}, G_2={(0, 2), (0, 3), (0, 4)} and G_3={(4, -3), (4, -4), (4, -5)}. We calculate the average distance within a group and the average distance between groups in Table I. The average distance within a group is generally very low, while the average distance between groups is relatively high, and as a result, motion vectors reveal a structure of clustering.

TABLE I. AVERAGE DISTANCE WITHIN A GROUP AND BETWEEN GROUPS

Group	Average Distance		
	G_1	G_2	G_3
G_1	0.45	13.94	19.13
G_2	-	0.56	5.19
G_3	-	-	0.59

The motion vectors of a frame in a normal video sequence usually have only several count peaks. Therefore, the motion vectors only have several clusters with high counts. The clustering statistics could be computed from the motion vectors of the past blocks. Based on these clustering statistics, several efficient predictors with most possibilities are provided for the motion estimation of the following blocks. It's usually much possible for one block to quickly find the best-matched candidate with the predictors.

III. PROGRESSIVE MOTION VECTOR CLUSTERING

Let D be a dataset of motion vectors and let p and q be two vectors in D. The Manhattan distance [11] between $p(x_p, y_p)$ and $q(x_q, y_q)$ is defined as $d(p, q)=|x_p-x_q|+|y_p-y_q|$. Based on the Manhattan distance, we define the reachable relationship between two motion vectors as well as the motion vector clusters. The reachable definition refers to those vectors p, q satisfying $d(p, q)\leqslant 2$.

Definition 1: (directly reachable) If $d(p, q)=1$, q is directly reachable from p, denoted by $p \to q$.

Definition 2: (indirectly reachable) If ① $d(p, q)=2$ and ② there exists a vector r in D subject to $p \to r$ and $r \to q$ i.e. $d(p, r)=1$ and $d(r, q)=1$, q is indirectly reachable from p, denoted by $p \succ q$.

Definition 3: (cluster) A cluster C_i is a non-empty subset of D satisfying the following conditions:

① $\forall q \in D$:if $q \to rep(C_i)$, then $q \in C_i$

② $\forall q \in D$:if $q \succ rep(C_i)$ and $q \not\to rep(C_j)\, j \neq i$ then $q \in C_i$

where *rep*() denotes the representative vector of a cluster.

According to the definition of cluster, a motion vector m is assigned to cluster C when the following rule of clustering is satisfied.

The Rule of Clustering: Let C_1, C_2 ... C_n be the existing clusters. A new motion vector m is assigned to a certain cluster C_i if m satisfies one of the following conditions:

① $m \to rep(C_i)$

② $m \succ rep(C_i)$ and for $\forall rep(C_j)$, $j \neq i$, $m \not\to rep(C_j)$

where *rep*() denotes the representative vector of a cluster.

The Rule of Clustering is illustrated in Fig. 2. A motion vector is denoted by an integral point in the coordinates (*x*, *y*), a motion vector cluster is denoted by a dashed circle, and the representative vector of a cluster is denoted by a shaded point. Suppose p, q, r, s, u and t are motion vectors and C_1 is the existing vector cluster. Initially, C_1 has only one member i.e. $C_1=\{p\}$ and $rep(C_1)=p$. Since $p \to q$, $q \to r$ and $q \to s$, vectors r and s are indirectly reachable from p. Vector q satisfies condition ① of the rule and therefore q is assigned to cluster C_1. Vectors r and s satisfy condition ② of the rule and thus r and s are also assigned to cluster C_1. Vectors u and t can not be assigned to cluster C_1, so new clusters $C_2=\{u\}$ and $C_3=\{t\}$ are created.

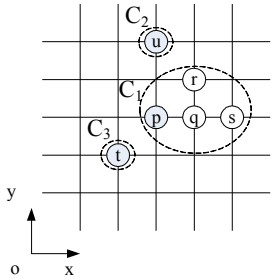

Figure 2. The clustering of motion vectors.

For a motion vector n which can not be assigned to any existing clusters, a new cluster $C_{new}=\{n\}$ with $rep(C_{new})=n$ is created. After the creation of C_{new}, some members of existing clusters C_1, C_2 ... C_n are assigned to C_{new} when the following rule is satisfied.

The Rule of Membership Change: Let motion vector m be a member of cluster C_i. When C_{new} is created, if m satisfies $m \succ rep(C_i)$ and $m \to rep(C_{new})$, assign m to C_{new}, where *rep*() denotes the representative vector of a cluster.

The change of the membership of the existing clusters is illustrated in Fig. 3. Suppose v is a motion vector and v can not be assigned to C_1, C_2 or C_3. Therefore, a new cluster $C_4=\{v\}$ is created and $rep(C_4)=v$. According to the Rule of Membership Change, vector s becomes a member of C_4.

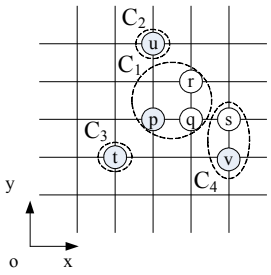

Figure 3. The change of membership of clusters.

The representative vector of a cluster will be re-selected because of the change of its members and the change of the counts of its members. The selection of the representative vector of a cluster should follow the rule as:

The Rule of Representative Vector Selection: Let C be a cluster and initially $rep(C)=p$. Vector p' is selected as the representative vector of cluster C if p' satisfies the following conditions:

① $p' \in \{p\} \cup \{q \mid q \in C, p \to q\}$

② $cost(C,p') = min\{cost(C,i) \mid i \in \{p\} \cup \{q \mid q \in C, p \to q\}\}$,

in which $cost(C, i)$ is defined as:

$$cost(C,i) = \frac{1}{\sum_{j \in C, d(i,j) \le 2} n(j)} \sum_{j \in C, d(i,j) \le 2} n(j) \cdot d(i,j) \qquad (1)$$

In (1), $n(j)$ denotes the count of vector j and $d(i, j)$ denotes the Manhattan distance between vector i and vector j. The $cost(C, i)$ indicates the average distance within cluster C when i is selected as the representative vector.

The Rule of Representative Vector Selection is illustrated in Fig. 4. Let the count of s be 1 and the count of v be 2 at first. Suppose the count of s increases to 8 and the count of v remains 2. According to (1), $cost(C_4, s)=0.2$ and $cost(C_4, v)=0.8$. From the Rule of Representative Vector Selection, s will become the representative vector because $cost(C_4, s) < cost(C_4, v)$.

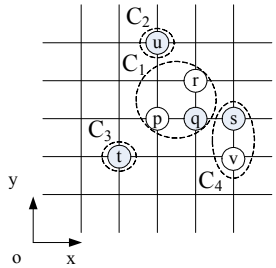

Figure 4. The selection of representative vectors.

The Progressive Clustering algorithm periodically assigns newly generated motion vectors to the existing clusters. Let $M=\{m_1, m_2 ... m_w\}$ be the set of newly generated motion vectors and $n'(m_j)$ denotes the count of m_j in M. Let $E=\{C_1, C_2 ... C_n\}$ be the set of existing clusters. The procedures of the Progressive Clustering algorithm are described as follows:

1. Adding new motion vectors to existing clusters.

(1) For each motion vector m_j in M, if m_j is already a member of cluster C_k ($m_j \in C_k$), increase the count of m_j in C_k by $n(m_j)=n(m_j)+n'(m_j)$, in which $n(m_j)$ denotes the count of m_j in C_k, and remove m_j from M.

(2) For each motion vector m_j in M, if m_j meets condition ① in the Rule of Clustering, assign m_j to the corresponding cluster and remove m_j from M.

(3) For each motion vector m_j in M, if m_j meets condition ② in the Rule of Clustering, assign m_j to the corresponding cluster and remove m_j from M.

After the adding above, generate $U=\{u_1, u_2 ... u_r\}$ as the set of remaining unassigned motion vectors of M, and set the number of algorithm iteration *num* as 0.

2. Clustering the remaining motion vectors.

For each unassigned motion vector u_j in U, create a new cluster $C'_j=\{u_j\}$, add C'_j to E by $E=E \cup C'_j$, and then check all the reachable vectors of u_j by the following (1) and (2).

(1) For each directly reachable vector p of u_j, if p is in U and p is an unassigned vector, assign p to C'_j according to the Rule of Clustering, otherwise, if p is a member of C_i and $p \succ rep(C_i)$, assign p to C'_j according to the Rule of Membership Change.

(2) For each indirectly reachable vector q of u_j, if q is in U and q is an unassigned vector, assign q to C'_j according to the Rule of Clustering.

3. If *num* is larger than a certain number (usually 10), the clustering algorithm terminates, otherwise, for each cluster C_i in E, reselect the representative vector of C_i according to the Rule of Representative Vector Selection.

4. Let $N=\{n_1, n_2 ... n_s\}$ be the set of members of those clusters whose representative vectors changed and set U as an empty set.

(1) For each motion vector n_k in N, if n_k meets condition ① in the Rule of Clustering, assign n_k to the corresponding cluster and remove n_k from N.

(2) For each motion vector n_k in N, if n_k meets condition ② in the Rule of Clustering, assign n_k to the corresponding cluster and remove n_k from N.

After (1) and (2), add the remaining motion vectors in N to U.

5. If U is an empty set, the clustering algorithm terminates, otherwise, increase *num* by *1* and go to step 2.

IV. CLUSTERING BASED SEARCH ALGORITHM

According to the spatial correlation within frames, the motion of a block has a big possibility to be close to that of some of its neighbors. We can infer that the motion vector of a block will belong to one of the clusters that hold the neighbor blocks' motion vectors. Therefore, the representative vectors of these clusters will be used as the predictors for the motion estimation of the block. With these predictors, the search range is narrowed into several small search areas and consequently the number of candidates to be searched is also reduced. Searching in the small search areas is probable to find the best-matched candidate, but under the circumstance that it is failed, searching in the entire search range should be performed.

The search process for the current block is illustrated in Fig. 5. Vector mc_L denotes the representative vector of the cluster to which the motion vector of the left neighboring block belongs, vector mc_U indicates the representative vector of the cluster to which the motion vector of the upper neighboring block belongs, and vector mc_{max} is the representative vector of the cluster with the highest count. Vectors mc_L, mc_U and mc_{max} are generated by the Progressive Clustering algorithm. The proposed search algorithm utilizes them as the motion vector predictors for the current block. The search process consists of two phases: (1) with these predictors, only the candidates in the three size 3×3 areas are searched by FS algorithm to find a local minimum block distortion (MBD) point, (2) if the distortion of the local MBD point is lower than a certain threshold, the displacement of the local MBD point is regarded as the motion vector of the current block, otherwise, candidates in the entire search range are searched without predictors by Line Search algorithm to obtain the motion vector of the current block.

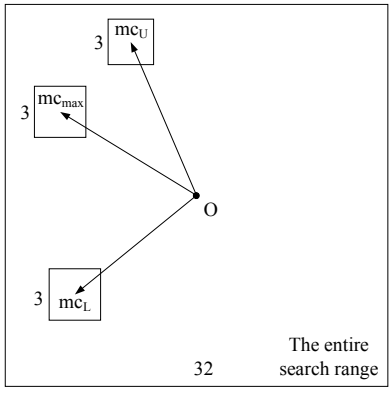

Figure 5. The search process of the current block.

For a video frame consisting of W×H blocks, blocks are grouped and motion estimation will be made group by group. When one group has been completed, the Progressive Clustering algorithm will be processed. Specific steps of the proposed motion estimation algorithm are described below:

1. Separate the blocks of a frame into W+H-1 groups by the diagonals. Blocks with the same number in Fig. 6 belong to one group.

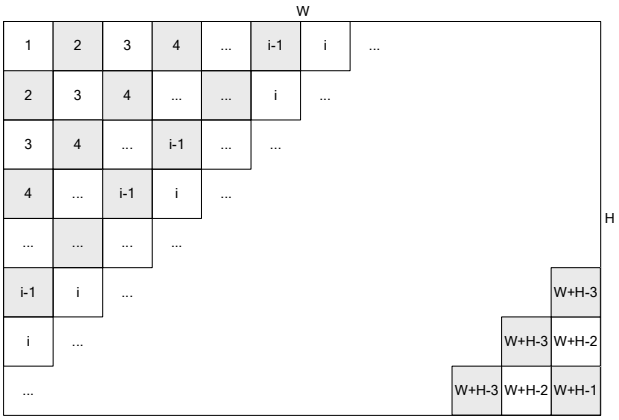

Figure 6. The blocks grouped by diagonals.

2. Choose the *i*th group of blocks and estimate the motion vector for each block in the *i*th group. For each block in the *i*th group, search the candidates in the three size 3×3 areas by FS algorithm to find the local MBD point with mc_L, mc_U and mc_{max} as the predictors, if the block distortion of the local MBD point is lower than the threshold, regard the displacement of the local MBD point as the motion vector, otherwise, search the candidates in the entire search range by Line Search algorithm to obtain the motion vector.

3. Assign the motion vectors of the *i*th block group to the existing vector clusters. Let $M=\{m_1, m_2 \ldots m_p\}$ be a set of the motion vectors of the *i*th block group and $E=\{C_1, C_2 \ldots C_n\}$ be a set of the existing clusters. M is assigned to E by the Progressive Clustering algorithm.

4. Check all the block groups. If there remain unsearched groups, $i=i+1$ and go to step 2, otherwise, report the motion vectors of the blocks in the frame.

In this way, the Clustering Based Search algorithm periodically invokes the Progressive Clustering algorithm to assign the motion vectors of a group of blocks to the existing clusters and then make clustering statistics. These clustering statistics are utilized as the vector predictors for the next group of blocks.

V. EXPERIMENT EVALUATION

To evaluate the performance of our algorithm, we apply it to five high resolution video sequences: Aspen (1080p), Blue sky (1080p), Park joy (720p), Ducks take off (720p) and In to tree (720p) [10]. In our experiment, the size of a block is 16×16 and the search range is (-16, 15). As 1080 is not completely divisible by 16, our experiment only takes the 1920×1072 pixels of the 1080p videos into account and the rest 8 rows of pixels are omitted. We use the sum of absolute difference (SAD) as the metric for block distortion. The block distortion threshold for our algorithm is 2048 and the size of small search areas in our algorithm is 3×3. We use 50 frames of each video sequences and test our algorithm on a

personal computer of Intel Core2 CPU E6750 at 2.66 GHz and 2G RAM.

We choose the mean-square error (MSE) as the criterion for measuring the performance of motion estimation algorithms. The MSE compares the motion compensated image frame with the original image frame. The lower the MSE, the smaller the energy of the prediction error and therefore the more effective the motion estimation algorithm is. We compare our algorithm with the FS, UMHexagonS, DS and PLS algorithms. Table II shows the MSE performance for each algorithm on the five test sequence. For sequences where only small motions are involved, such as Aspen, the MSE performance of these algorithms is very close. However, for the sequences with large motions, such as Blue sky and Park joy, our algorithm outperforms UMHexagonS, DS and PLS. The average MSE value of our algorithm is slightly higher than the value of FS and is lower than those of UMHexagonS, DS and PLS.

TABLE II. MSE PERFORMANCE COMPARISON

Video Sequence	MSE				
	Our algorithm	***FS***	***UMHexagonS***	***DS***	***PLS***
Aspen	21.1	18.24	21.18	23.15	19.15
Blue sky	32.16	26.46	51.27	77.22	33.36
Park joy	283.58	269.8	336.31	562.57	344.39
Ducks take off	102.83	102	102.31	134.33	103.56
In to tree	35.64	31.39	32.79	84.14	32.13
Average	95.062	89.58	108.77	176.28	106.52

We also do the frame-by-frame comparison of our algorithm with the FS, UMHexagonS, DS and PLS algorithms. Fig. 7 and Fig. 8 show the MSE measure versus frame number for Blue sky and Park joy sequences. As shown in these figures, the MSE values of our algorithm stay very close to those of FS only with small deviations when the sequences involves large motions. However, the MSE values of UMHexagonS and DS rise obviously for the sequences with high motions. These figures also show that the MSE curve of our algorithm is more approximate to the curve of FS than that of PLS.

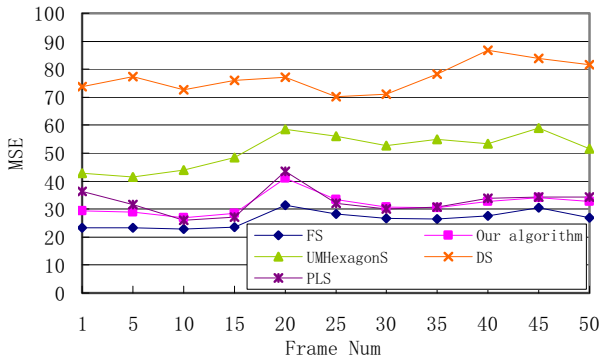

Figure 7. The frame-by-frame comparison of Blue sky sequence.

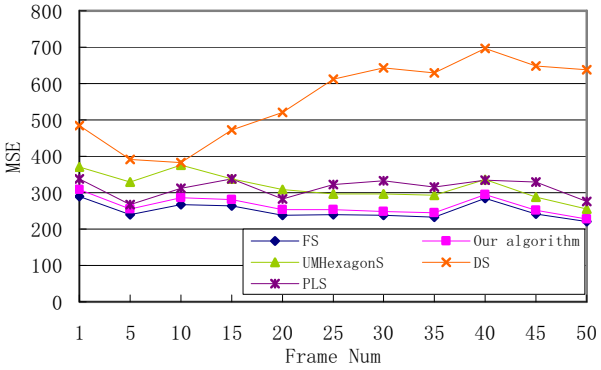

Figure 8. The frame-by-frame comparison of Park joy sequence.

For complexity comparison, we select the time cost per frame as the metric. Our algorithm is compared to the FS, UMHexagonS, DS and PLS algorithms in Table Ⅲ. The speed of our algorithm is as fast as the DS algorithm while the MSE of our algorithm is much lower. Our algorithm performs 4 times faster than the UMHexagonS and PLS algorithms. Compared to FS, the speedup ratio of our algorithm is nearly 35 times in average.

TABLE III. COMPARISON ON THE TIME COST OF MOTION ESTIMATION

Video Sequence	Time Cost (ms/frame)				
	Our algorithm	***FS***	***UMHexagonS***	***DS***	***PLS***
Aspen	77.78	3283	368.67	75.86	399.76
Blue sky	77.87	3255	355.45	116.55	381.35
Park joy	89.96	1460	171.67	54.12	183.34
Ducks take off	47.49	1440	127.78	25.86	139.39
In to tree	29.89	1445	145.65	28.39	152.86
Average	64.59	2176.6	233.84	60.16	251.34

From these experiment results, it is obviously that our algorithm has the capability to reduce the large computational burden of FS algorithm with negligible increase in the MSE performance. Our algorithm has an effective and efficient motion estimation performance due to the clustering statistics of motion vectors. The overhead of motion vector clustering is measured by the time cost of clustering. As shown in Table Ⅳ, the cost of clustering only occupies 4.5% of the total cost of motion estimation in average. As a result, the overhead of vector clustering is very low.

TABLE IV. TIME COST OF CLUSTERING

Video Sequence	Time Cost (ms/ frame)		
	Total motion estimation	*Clustering*	*Percentage*
Aspen	77.78	5.4	6.94%
Blue sky	77.87	4.9	6.29%
Park joy	89.96	2.1	2.33%
Ducks take off	47.49	0.9	1.90%
In to tree	29.89	1.4	4.68%
Average	64.59	2.94	4.55%

Our algorithm groups blocks by diagonals, and the motion estimation of one block is independent to the others' in the same group. Hence the blocks in a group could be processed in parallel without additional efforts. We also implement our algorithm on a NVIDIA 8800GTX graphics card with CUDA. CPU makes progressive clustering statistics, while GPU executes motion estimation for blocks. As shown in Table Ⅴ, the preliminary results on the GPU-based implementation get around 4 times faster.

TABLE V. COMPARISON ON THE TIME COST OF CPU AND GPU ACCELERATION

Video Sequence	Time Cost (ms/ frame)		
	CPU only	*CPU with GPU acceleration*	*Speedup ratio*
Aspen	77.78	20.1	3.9
Blue sky	77.87	18.9	4.1
Park joy	89.96	29.2	3.1
Ducks take off	47.49	11.3	4.2
In to tree	29.89	7.6	3.9
Average	64.59	17.4	3.8

VI. CONCLUSION AND FUTURE WORK

A novel fast motion estimation algorithm, Clustering Based Search, is described in this paper. The main features of our algorithm are counting the motion vectors of past blocks to make clustering statistics and then utilizing the clusters to provide efficient predictors with most possibilities for the following blocks. It is usually probable for one block to rapidly find the best-matched candidate with the predictors. From the experiment results, the MSE performance of our Clustering Based Search algorithm is very close to that of the FS algorithm while its speed is nearly 35 times faster. It is also shown that our algorithm outperforms the UMHexagonS, DS and PLS algorithms, especially for video sequences with large motions.

Our algorithm is preliminarily implemented on GPU with CUDA. The speedup of the GPU-based implementation is around 4 times. To develop the algorithm for variable block size motion estimation and fractional pixel refinement may be promising. We would also implement our algorithm on a video codec system with H.264/AVC.

ACKNOWLEDGMENT

This work is supported by the National 863 Program of China under Grant No.2012AA011801, the Natural Science Foundation of China under Grant No.61170188, the National 973 Program of China under Grant No. 2009CB320805, and Fundamental Research Funds for the Central Universities of China.

REFERENCES

[1] T. Koga, K. Linuma, A. Hirano, Y. Iijima, T.Ishiguro, "Motion-compensated interframe coding for video conferencing," Proc. Nat. Telecommunication Conference, New Orleans, Dec. 1981, pp. G5.3.1-G5.3.5.

[2] R. Li, B. Zeng, ML Liou, "A new three-step search algorithm for block motion estimation," IEEE Transactions on Circuits and Systems for Video Technology, vol. 4, no. 4, Aug. 1994, pp. 438-442, doi:10.1109/76.313138.

[3] L.M. Po, W.C. Ma, "A novel four-step search algorithm for fast block motion estimation," IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, no. 3, Jun. 1996, pp. 313-317, doi:10.1109/76.499840.

[4] S. Zhu, K.K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," IEEE Transactions on Image Processing, vol. 9, no. 2, Feb. 2000, pp. 287-290, doi:10.1109/83.821744.

[5] C. Zhu,X. Lin, L.P. Chau, KP Lim, HA Ang, CY Ong, "A novel hexagon-based search algorithm for fast block motion estimation," Proc. 2001 IEEE International Conference on Acoustics, Speech and Signal Processing, Salt Lake City, May 2001, pp. 1593-1596, doi:10.1109/ICASSP.2001.941239.

[6] Y.W. Huang, S.Y. Ma, C.F. Shen, L.G. Chen, "Predictive Line Search: an efficient motion estimation algorithm for MPEG-4 encoding systems on multimedia Processors," IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 1, Jan. 2003, pp. 111-117, doi:10.1109/TCSVT.2002.808093.

[7] C. Zhibo, P. Zhou, H. Yun, and Y. Chen, "Fast integer pel and fractional pel motion estimation for JVT," presented at the 6th Meeting of JVT-F017, Awaji, JP, 2002.

[8] Z.R. Shi, W.A.C. Fernando, and D.V.S.X. De Silva, "A motion estimation algorithm based on Predictive Intensive Direction Search for H.264/AVC," Proc. 2010 IEEE International Conference on Multimedia and Expo (ICME), Suntec City, July 2010, pp. 667-672, doi:10.1109/ICME.2010.5582997.

[9] Z.R. Shi, W.A.C. Fernando, "Adaptive Direction Search Algorithms based on Motion Correlation for Block Motion Estimation," IEEE Transactions on Consumer Electronics, Vol. 57, No. 3, Aug. 2011, pp. 1354-1361, doi:10.1109/TCE.2011.6018894.

[10] http://media.xiph.org/video/derf/.

[11] E.F. Krause, "Taxicab Geometry: An adventure in non-Euclidean geometry," NY:Dover Publications, 1987.