Progressive Motion Vector Clustering for Motion Estimation and Auxiliary Tracking

KE CHEN, ZHONG ZHOU, and WEI WU, Beihang University

The motion vector similarity between neighboring blocks is widely used in motion estimation algorithms. However, for nonneighboring blocks, they may also have similar motions due to close depths or belonging to the same object inside the scene. Therefore, the motion vectors usually have several kinds of patterns, which reveal a clustering structure. In this article, we propose a progressive clustering algorithm, which periodically counts the motion vectors of the past blocks to make incremental clustering statistics. These statistics are used as the motion vector predictors for the following blocks. It is proved to be much more efficient for one block to find the best-matching candidate with the predictors. We also design the clustering based search with CUDA for GPU acceleration. Another interesting application of the clustering statistics is persistent static object tracking. Based on the statistics, several auxiliary tracking areas are created to guide the object tracking. Even when the target object has significant changes in appearance or it disappears occasionally, its position still can be predicted. The experiments on Xiph.org Video Test Media dataset illustrate that our clustering based search algorithm outperforms the mainstream and some state-of-the-art motion estimation algorithms. It is 33 times faster on average than the full search algorithm with only slightly higher mean-square error values in the experiments. The tracking results show that the auxiliary tracking areas help to locate the target object effectively.

Categories and Subject Descriptors: H.4.m [**Information Systems Applications**]: Miscellaneous; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering*

General Terms: Algorithms

Additional Key Words and Phrases: Progressive clustering, block matching, motion estimation, auxiliary tracking

ACM Reference Format:
Ke Chen, Zhong Zhou, and Wei Wu. 2015. Progressive motion vector clustering for motion estimation and auxiliary tracking. ACM Trans. Multimedia Comput. Commun. Appl. 11, 3, Article 33 (January 2015), 23 pages.
DOI: http://dx.doi.org/10.1145/2700296

1. INTRODUCTION

In most existing international video standards, such as the ISO MPEG series and the ITU-T H.26X series, motion estimation has been adopted to remove temporal redundancy within frames and provide coding systems with high compression ratio. The simplest and most effective method for motion estimation is the full search (FS) algorithm. It compares all the candidates in the search area and finds the best-matching block. FS is the one with the minimum error but always with the highest computation due to the thorough candidate matching.

This work is supported by the National 863 Programs of China under Grant No. 2012AA011801 and No. 2012AA011803 and the National Natural Science Foundation of China under Grant No. 61170188.
Corresponding author's address: Z. Zhou, No. 37 Xueyuan Road, Haidian District, Beijing, P.R. China 100191; email: zz@buaa.edu.cn.

DOI: http://dx.doi.org/10.1145/2700296

In order to reduce the searched candidates, many fast motion estimation algorithms have been proposed. These algorithms apply different search patterns, search strategies and motion vector predictors. They reduce the search computation, but the search accuracy and the speed always ask for a trade-off. The motion vector similarity between neighboring blocks, that is, the spatial correlation of motion vectors, is widely used by these fast algorithms. However, those blocks that are not neighbors may also have similar motions, due to close depths or belonging to the same object inside the scene. The motion vectors usually have several kinds of patterns, and these patterns reveal a clustering structure.

We take two adjacent frames in video sequence *Park Joy* (720p) [Xiph.org 2013] as an example to illustrate the clustering features of motion vectors. Figure 1(a) shows two adjacent frames of the *Park Joy* sequence. Figure 1(b) shows the histogram of motion vectors computed by the 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 block count axis shows the number of the blocks whose motion vectors are (x, y). We can see that the motion vectors follow a two-dimensional discrete distribution, and they have several peaks, that is, the counts of some vectors and their neighbors are much higher than the others.

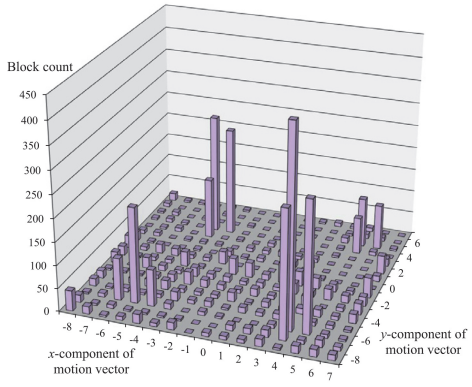
The motion vectors of the frame have a structure of clustering. To illustrate it, we manually select 20 different motion vectors, whose counts are more than 40, as the cluster centers. Then, the K-means [MacQueen 1967] algorithm is employed to classify the motion vectors into 20 clusters. Figure 1(c) gives the clustering results of K-means. We use the same color to illustrate the blocks whose motion vectors belong to the same cluster. As shown in Figure 1(c), due to the spatial correlation of block motion, neighboring blocks usually belong to the same cluster. More importantly, the nonneighboring blocks might also be classified into the same cluster, because these blocks might also have similar motions. We calculate the blocks within each cluster and sort these clusters by the block count. Figure 1(d) shows the block counts of the top 10 clusters. We can see that the motion vectors only have several clusters with high counts. With the help of these clusters, it is possible to only search a few candidates to find the best-matching block. It can be inferred that the results could be better with more accurate cluster centers instead of manual selection. Then we can draw a conclusion that, besides the spatial correlation, it is promising to explore the clustering statistics of motion vectors.

In this article, we propose a progressive motion vector clustering algorithm. It periodically counts the motion vectors of the past blocks to make incremental clustering statistics. To the best of our knowledge, little work has been done before on the clustering statistics of motion vectors for block matching. Our clustering-based search integrates the spatial correlation and the clustering statistics of motion vectors. It uses the clustering information of neighboring blocks to select predictors for block matching. It is usually much more possible for one block to quickly find the best-matching candidate with the clustering based predictors.

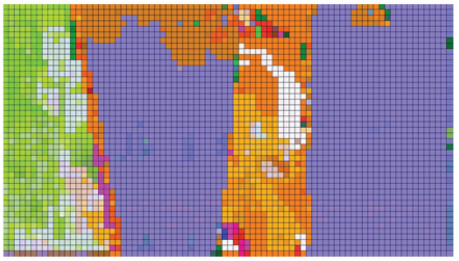
Another interesting application of the clustering statistics is the persistent tracking for static objects, especially for the plate-shaped objects, such as traffic signs and advertisement billboards. These statistics are useful for estimating the position of a static target object. When the target disappears occasionally or its appearance changes significantly due to camera movement, the position of the target still can be estimated. Figure 2 shows the appearance changes of a road sign in a video sequence.

We count the pixels of the road sign and calculate its average YUV values. Figure 3 shows the number of pixels of the traffic sign in each frame. The pixels decrease in the first 50 frames, reach the minimum at the 50th frame and then increase gradually. When the pixels are quite few, a tracker usually fails to track and loses the target. It is also very difficult for the tracker to recover due to severe changes of the target in color

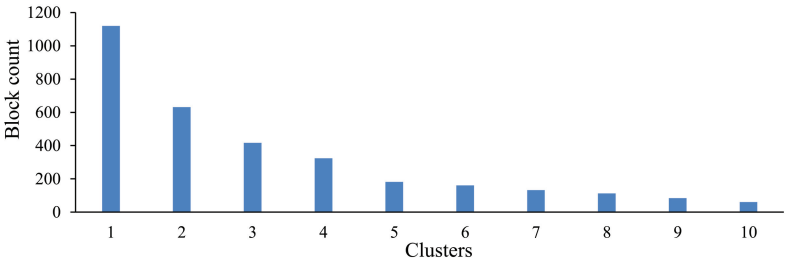
(a) Two adjacent frames of the *Park Joy* sequence [Xiph.org 2013].



(b) The histogram of motion vectors.



(c) The clustering results of motion vectors. The blocks whose motion vectors belong to the same cluster are painted with the same color.



(d) The block counts of the top 10 clusters. The first six clusters occupy nearly 80% blocks.

Fig. 1. The clustering statistics of motion vectors for adjacent frames.

(a) Frame #2. (b) Frame #51. (c) Frame #85.

Fig. 2. The appearance of the road sign changes in different frames.

appearance. Figure 4 shows the average YUV values of the road sign for each frame. The change of the color appearance is discontinuous, and the sharp breaks occur from the 45th frame to the 55th. The disappearances and the sudden appearance changes make difficulties in persistent object tracking.

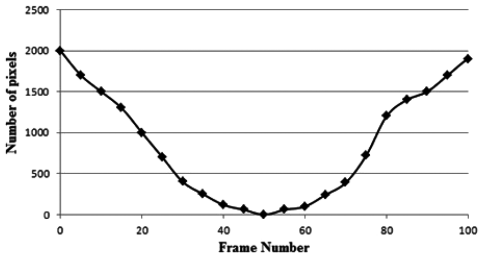


Fig. 3. The number of pixels of the road sign.

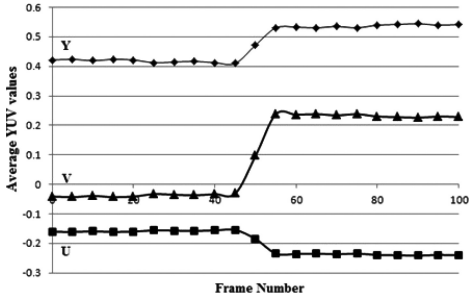


Fig. 4. The average YUV values of the road sign.

(a) Frame #2. (b) Frame #51. (c) Frame #85.

Fig. 5. Tracking an object persistently with the auxiliary tracking areas (red: auxiliary tracking areas, yellow: tracking results).

However, static objects at close depths usually have similar motions, and the position of the target can be roughly estimated by the others. Inspired by this, we create several auxiliary tracking areas from the motion vector clustering statistics. The creation does not need 3D calculation or depth analysis. As Figure 5 shows, with the help of the auxiliary tracking areas, the target object (road sign) could be tracked effectively despite severe appearance changes or even occasional disappearances.

This article is an extension of our conference paper [Chen et al. 2012] which focuses on the clustering based search for motion estimation. In this article, we give a systematic description of motion vector clustering and propose a progressive clustering algorithm. We also extend our clustering based search and design its GPU-based algorithm. In our other work [Zhou et al. 2012], we use the average motion vector of the target object to create auxiliary tracking areas. Instead of the simple average motion vector, this article improves the auxiliary tracking area creation by the clustering statistics.

The rest of this article is organized as follows. Section 2 reviews prior work. Section 3 gives the clustering definition and the proposed motion vector clustering algorithm. Section 4 introduces the clustering based search and its GPU-based algorithm. Section 5 presents the creation of auxiliary tracking areas. Our experiment results are illustrated in Section 6, and Section 7 draws a conclusion.

2. RELATED WORK

2.1. Motion Estimation

Block matching motion estimation (BMME) divides the current frame into non-overlapping blocks of size N $\times$ N, and each block is designated to a search area in the reference frame. Then, each block is matched with the candidate blocks in the search area to find the best-matching candidate. The displacement of the best-matching block is the motion vector of the current block. The most straightforward method to find

the best-matching block is the full search (FS) algorithm. It matches all the candidate blocks in the search area to find the global optimal candidate.

The FS algorithm comes up with the global minimum distortion block, but its computation is exhaustive due to the thorough candidate matching. To overcome the computation drawback, numerous fast motion estimation algorithms have been proposed, which try to achieve what the FS algorithm does but with less computational efforts. They apply different search patterns and search strategies to reduce the searched candidates, such as three-step search (TSS) [Koga et al. 1981], new three-step search (NTSS) [Li et al. 1994], four-step search (FSS) [Po and Ma 1996], diamond search (DS) [Zhu and Ma 2000], hexagon search (HS) [Zhu et al. 2001] and iterative random search (IRS) [Porto et al. 2013]. However, their accuracies inevitably degrade, especially in video shots with large motions.

Fast motion estimation algorithms also use motion vector predictors to narrow the search range and reduce the computation. Due to the spatial correlation, the motion vector of a block could be predicted by those of its neighbors. Based on this, some predictive motion estimation algorithms have been put forward, such as hybrid unsymmetrical cross multi-hexagon grid search (UMHS) [Chen et al. 2002], predictive line search (PLS) [Huang et al. 2003], predictive intensive direction searching (PIDS) [Shi et al. 2010], and simulated annealing adaptive search (SAAS) [Shi et al. 2011]. These predictive algorithms use the motion vectors of the spatial/temporal neighboring blocks to build initial predictors. They could narrow the search area as well as reduce the computation, but the search accuracy and the speed always ask for a tradeoff. In fact, more than the spatial correlation, those nonneighboring blocks may also have similar motions, because they may have close depths or belong to the same object.

The 2D motion analysis has already received some attention in image segmentation [Hennebert et al. 1996; Cucchiara et al. 2003]. The region-level motion-based graph representation of the image partition is presented by Gelgon and Bouthemy [2000]. However, the block-based motion estimation usually has the restriction of independent block matching to keep parallel possibilities, and thus avoids global optimization. Our method is proposed on the basis of the prevailing block matching method for the video coding standards. In this article, we investigate the clustering statistics on the past motion vectors to provide effective predictors for the following blocks. Its computation is quite fast and easy to be implemented.

2.2. Object Tracking

Object tracking is defined as a problem of estimating the trajectory of an object in the image plane [Yilmaz et al. 2006]. In other words, a tracker is to repeatedly locate the tracked object in successive frames. There are numerous tracking approaches, such as the mean-shift tracking [Comaniciu et al. 2000], the layering tracking [Tao et al. 2002], the SVM tracking [Avidan 2004] and so on.

Persistent Tracking is a challenging task due to severe appearance changes or even occasional disappearances of the target. To deal with the problem of appearance changes, some object tracking methods have been proposed. Ross et al. [2008] present a tracking method that incrementally learns and updates a low-dimensional subspace representation of the target object during visual tracking. Han et al. [2007] propose a probabilistic sensor selection method and apply a mixture of kernel-based Bayesian filters to object tracking. Du and Piater [2008] integrate multiple cues by Linked Hidden Markov Models and design a Sequential Auxiliary Particle Belief Propagation algorithm to track target objects. Kwon and Lee [2010] propose a visual tracking decomposition scheme for multiple observation and motion models as well as trackers. Besides the algorithms for appearance changes, there are also some other tracking methods to solve transient disappearances and occlusions. Comaniciu and Ramesh

[2000] use a Kalman Filter to estimate the motion parameters of a target object in the future frames. Okuma et al. [2004] develop a boosted particle filter that combines mixture particle filters and the Adaboost algorithm. Yin and Collins [2008] solve the problem of object detection via Adaptive Simulated Annealing and propose a tracking method which can recover from some inevitable tracking failures. Wu et al. [2012] propose covariance matching for partial differential equation-based (PDE-based) contour tracking.

Although these tracking methods can deal with appearance changes and transient occlusions, they are apt to fail for the long period of object tracking, since the target may not only significantly change the appearance, but also be minimized or even disappear occasionally during its display lifetime. For a long period of disappearance, the target object would probably move in a different way that is not modeled by these tracking methods. In view of these factors, we make the immediate application of the motion vector clustering to tracking. It is not only relying on the color or motion features, but also some external static image partitions. This will help in such situations as occasional disappearances and significant appearance changes.

3. PROGRESSIVE MOTION VECTOR CLUSTERING

3.1. Definition on Reachability Based Cluster

Let D be a dataset of motion vectors, and the motion vector clustering $\Theta = \{C_1, C_2 \dots C_n\}$ is a partition of D, which separates the motion vectors into multiple disjoint clusters based on some distance metrics. We choose the Manhattan distance [Krause 1987] as the distance metric for clustering. The Manhattan distance is defined as $d(p, q) = |x_p - x_q| + |y_p - y_q|$, where x and y respectively denote the horizontal and vertical components of a motion vector. According to the Manhattan distance, the representative vector r_{C_i} of cluster C_i is the vector with the minimum average distance within the cluster.

The representative vector substitutes the other vectors within the cluster as the predictor for motion estimation. The search window predicted by the representative vector should overlap the search windows predicted by the other vectors as much as possible. In consideration of search window overlapping, we define the reachable relationships between two motion vectors p and q satisfying $d(p, q) \leq 2$. Based on the reachable relationships, the motion vector clusters are defined.

Definition 3.1 (*Directly Reachable*). If $d(p, q) = 1$, q is directly reachable from p, denoted by $p \rightarrow q$.

Definition 3.2 (*Indirectly Reachable*). If $d(p, q) = 2$ and there exists a vector s in D subject to $p \rightarrow s$ and $s \rightarrow q$, that is, $d(p, s) = 1$ and $d(s, q) = 1$, q is indirectly reachable from p, denoted by $p \succ q$.

Definition 3.3 (*Cluster*). A cluster C_i is a nonempty subset of D satisfying the following conditions:

(1) $\forall q \in D$: if $q \rightarrow r_{C_i}$, then $q \in C_i$;
(2) $\forall q \in D$: if $q \succ r_{C_i} (\exists p \in C_i, q \rightarrow p, p \rightarrow r_{C_i})$ and $q \nrightarrow r_{C_j} (j \neq i)$, then $q \in C_i$,

where r_{C_i} and r_{C_j} denote the representative vectors of cluster C_i and cluster C_j, respectively.

The direct reachability, indirect reachability and clusters are illustrated in Figure 6. The distance between p and s is 1, that is, $s \rightarrow p$. Similarly, $q \rightarrow s$. Because $q \rightarrow s$ and $s \rightarrow p$, p is indirectly reachable from q, that is, $q \succ p$. Although the distance between u and q is 2, u is not indirectly reachable from q. Let p and u be the representative

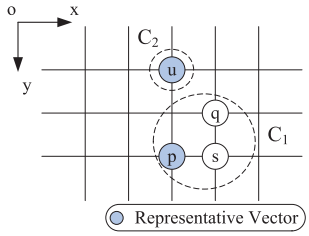


Fig. 6. The direct reachability, indirect reachability and clusters.

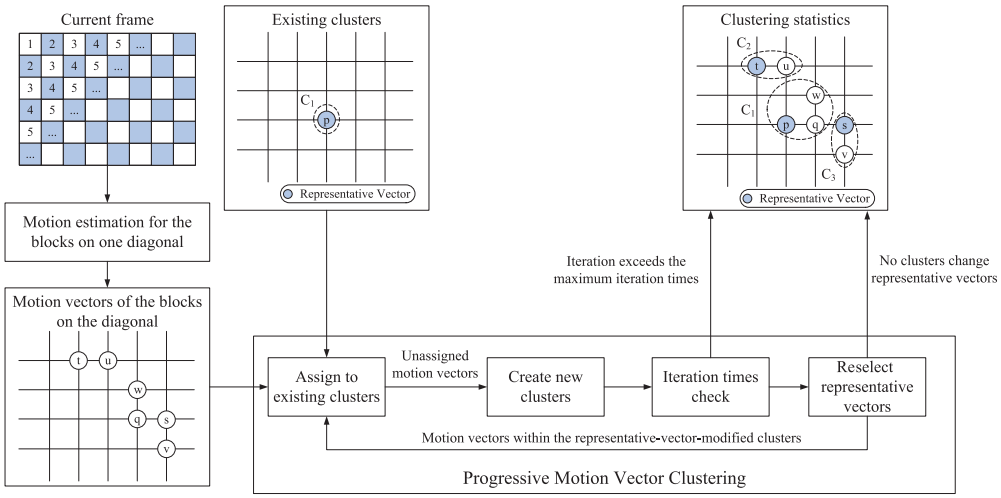


Fig. 7. The process of the progressive motion vector clustering.

vectors of clusters C_1 and C_2 respectively. Then, $s \in C_1$ because $s \rightarrow p$, and $q \in C_1$ because $q \succ p$ through s and $q \nrightarrow u$.

3.2. Our Clustering Algorithm

We divide the blocks of a frame into groups by the diagonals, and conduct motion estimation diagonal by diagonal. As Figure 7 shows, when the blocks on one diagonal are finished with their motion estimation, the progressive motion vector clustering is invoked to make the next incremental clustering statistics. It assigns the motion vectors of the blocks on the current diagonal to the existing clusters, rather than generates a completely new clustering structure. The clustering algorithm first assigns the motion vectors to the existing clusters, and then creates several new clusters to accommodate the remaining unassigned vectors. After that, it reselects the representative vector for each cluster, and then a new loop begins. The clustering algorithm reassigns the members of those clusters whose representative vectors are modified. Some of the members may not be reassigned to any clusters and become unassigned motion vectors. In this way, the algorithm repeats the assigning, creating and reselecting processes several times to get the final clustering result.

Let $M = \{m_1, m_2 \ldots m_w\}$ be the set of motion vectors to be clustered, and $E = \{C_1, C_2 \ldots C_n\}$ be the set of existing clusters. At the beginning of the first iteration, M is initialized as the motion vector set of the blocks on the current diagonal, by aggregating the blocks whose motion vectors are the same. Each m_j in M is a unique motion vector, and the count of m_j in M (denoted by n_{m_j}) is the number of the current

estimated blocks whose motion vectors are m_j. In the end of this iteration, some of the representative vectors might change. Then, we set M be the set of the motion vectors within the representative-vector-modified clusters, excluding those ones in the clusters whose representative vectors have no change. The proposed clustering algorithm makes clustering statistics by repeating the following processes several times.

(1) Assigning to existing clusters. The motion vectors in M are assigned to the existing clusters by the Rule of Directly Reachable Clustering and the Rule of Indirectly Reachable Clustering as follows. Only in the first iteration, the algorithm needs to check whether a motion vector m_j in M is already a member of cluster C. If so, it removes m_j from M and simply increases the count of m_j in C (denoted by N_{m_j}) by n_{m_j}, that is, $N_{m_j} = N_{m_j} + n_{m_j}$.

The Rule of Directly Reachable Clustering. An unassigned motion vector m is assigned to a cluster C_i if m satisfies $m \rightarrow r_{C_i}$.

The Rule of Indirectly Reachable Clustering. An unassigned motion vector m is assigned to a cluster C_i if $m \succ r_{C_i}$, $\exists\, q \in C_i\ (m \rightarrow q, q \rightarrow r_{C_i})$ and for $\forall r_{C_j}, m \nrightarrow r_{C_j} (j \neq i)$.

The Rule of Directly Reachable Clustering is prior to the Rule of Indirectly Reachable Clustering. If a motion vector satisfies the same clustering rule for several clusters, the motion vector is assigned to the cluster with the largest count. The clustering algorithm first checks the Rule of Directly Reachable Clustering for all the motion vectors in M, and assigns the qualified vectors to the corresponding clusters. Then, it checks the Rule of Indirectly Reachable Clustering for the remaining ones. According to these two rules, the algorithm assigns the motion vectors in M to the clusters in E.

(2) Creating new clusters. In some cases, not all the vectors in M could be assigned. Let U be the set of the unassigned vectors in M. The vectors in U are sorted by their counts in descending order. The proposed algorithm selects the first element u_{1st} of U and create a new cluster $C_{new} = \{u_{1st}\}$ with $r_{C_{new}} = u_{1st}$. Then, it checks the rest of U by the Rule of Directly Reachable Clustering and the Rule of Indirectly Reachable Clustering, and finds the vectors that could be assigned to C_{new}. After that, C_{new} is added to E and the members of C_{new} are removed from U. The new cluster creation is repeated several times until all the vectors in U are assigned.

The creation of a new cluster may change the membership of some indirectly reachable members within the existing clusters. The membership change makes these vectors become the directly reachable members of the new cluster. When a new cluster is created, the members of the existing clusters are tested by following rule, and the qualified vectors are assigned to the new cluster.

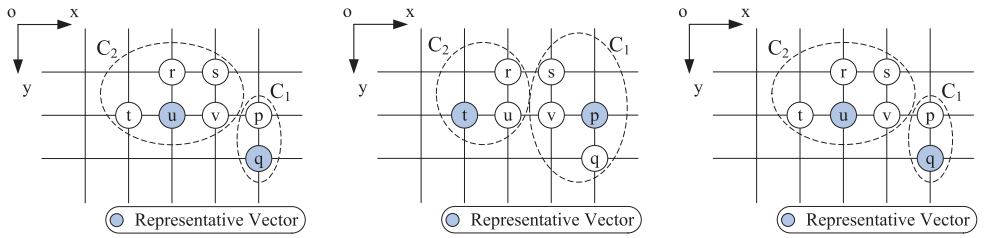
The Rule of Membership Change. Let m be a member of cluster C_i, and C_{new} be a new cluster. If m satisfies $m \succ r_{C_i}$ and $m \rightarrow r_{C_{new}}$, m is reassigned to C_{new}.

The membership change reduces the distance between a motion vector and its representative from 2 to 1. Therefore, the search window predicted by the new representative vector could cover more area of the window predicted by the motion vector.

(3) Reselecting representative vectors. The representative vector of a cluster is selected from the ones which are directly reachable from the previous representative vector and generate the minimum average distance within the cluster. Its selection meets the rule as follows.

The Rule of Representative Vector Selection. Vector p is selected as the representative vector of cluster C if $p = arg\,min_i\{cost(C, i) | i \in \{r_C\} \cup \{q | q \in C, r_C \rightarrow q\}\}$ in which $cost(C, i)$ is defined as:

$$cost(C, i) = \frac{1}{\sum_{j \in C, d(i,j) \leq 2} N_j} \cdot \sum_{j \in C, d(i,j) \leq 2} N_j \cdot d(i, j). \tag{1}$$



(a) Initial clusters. (b) After the first iteration. (c) After the second iteration.

Fig. 8. A specific nonconvergence case of our clustering algorithm.

In Equation (1), N_j denotes the count of vector j and $d(i, j)$ denotes the Manhattan distance between vector i and vector j. $cost(C, i)$ indicates the average distance within cluster C when i is selected as the representative vector. The selection of representative vectors may cause some vectors to become unassigned vectors, because the distances between these vectors and their new representatives become more than 2. In order to limit the number of the unassigned vectors, we restrict the candidate vectors within the ones directly reachable from the previous representative.

Our clustering algorithm reassigns the members of those clusters whose representative vectors are modified. It repeats the given processes several times to get the final results. We summarize the clustering algorithm and its pseudocode is described as Algorithm 1. We also give an example to illustrate the clustering procedure in the online Appendix A.

3.3. Discussion

In most cases, our clustering algorithm converges quickly after several iterations of assigning, creating and reselecting processes. However, in a few cases, the clustering algorithm could not converge. The main reason is that several motion vectors affect the balance of some clusters. They will be assigned to different clusters alternatively in each iteration. Figure 8 is a specific case to illustrate the nonconvergence of the clustering algorithm. Let C_1 and C_2 denote two motion vector clusters, and $\{p, q, r, s, t, u, \mathrm{v}\}$ denote motion vectors. The counts of motion vectors p, q, r, s, t, u and v are 9, 8, 2, 6, 3, 3 and 2 respectively. Initially, $C_1 = \{p, q\}$ with $r_{C_1} = q$ and $C_2 = \{r, s, t, u, \mathrm{v}\}$ with $r_{C_2} = u$. In the first iteration, p becomes the representative vector of C_1 because $cost(C_1, p) < cost(C_1, q)$ ($cost(C_1, p) = 0.47$ and $cost(C_1, q) = 0.53$). Similarly, t becomes the representative vector of C_2. The other motion vectors $\{q, r, s, u, v\}$ are reassigned to C_1 and C_2. After the first iteration, $C_1 = \{p, q, s, v\}$ with $r_{C_1} = p$ and $C_2 = \{r, t, u\}$ with $r_{C_2} = t$. In the second iteration, q is selected as the representative vector of C_1, because $cost(C_1, q) = 0.53$ is smaller than $cost(C_1, p) = 0.88$ and $cost(C_1, v) = 1.24$. For the same reason, u is selected as the representative vector of C_2. Then, motion vectors $\{p, r, s, t, \mathrm{v}\}$ are reassigned to C_1 and C_2. After the second iteration, $C_1 = \{p, q\}$ with $r_{C_1} = q$ and $C_2 = \{r, s, t, u, \mathrm{v}\}$ with $r_{C_2} = u$. Clusters C_1 and C_2 return to their initial status.

To evaluate the stability of the clusters, we measure the percentage of the clusters whose representative vectors are modified. In the evaluation, we use five test video sequences (*Aspen*, *Blue Sky*, *Park Joy*, *Ducks Take Off* and *In To Tree* [Xiph.org 2013]), and calculate the percentage of the representative-vector-modified clusters. Figure 9 illustrates the average percentage of the representative-vector-modified clusters versus iteration times. With the times of iteration increasing, the percentage of the representative-vector-modified clusters significantly decreases. When the iteration times are more than 5, the average percentage of the representative-vector-modified

ALGORITHM 1: Progressive Motion Vector Clustering Algorithm

Input: The motion vector set of the blocks on the current diagonal ($M = \{m_1, m_2 \dots m_w\}$), and the set of existing clusters ($E = \{C_1, C_2 \dots C_n\}$).
Output: The motion vectors in M are assigned to the clusters in E.
// Initialization
for each motion vector m_j in M **do**
  if $m_j \in C_k$ **then** // m_j is already a member of C_k
    $N_{m_j} = N_{m_j} + n_{m_j}$;
    $M = M - \{m_j\}$;
  end
end
// Iteration
$iteration\_times = 0$;
do
  // Assign to existing clusters
  DirectlyReachableClustering(M, E);
  IndirectlyReachableClustering(M, E);
  $U =$ the unassigned motion vectors in M;
  // Create new clusters
  SortbyCount(U);
  while $U \neq \varnothing$ **do**
    $u_{1st} =$ the first element of U;
    $C_{new} = \{u_{1st}\}$;
    $U = U - \{u_{1st}\}$;
    DirectlyReachableClustering(U, C_{new});
    IndirectlyReachableClustering(U, C_{new});
    MembershipChange(E, C_{new});
    $E = E \cup C_{new}$;
  end
  // Iteration times check
  if $iteration\_times \geq T_1$ **then** *return* **end**; //Usually $T_1 = 5$
  // Reselect representative vectors
  ReselectRepresentativeVector(E);
  $M =$ the set of the motion vectors within the representative-vector-modified clusters;
  $iteration\_times = iteration\_times + 1$;
while $M \neq \varnothing$

clusters is nearly 0.5%. We restrict the maximum iteration times of our clustering algorithm within 5 (i.e., $T_1 = 5$ in Algorithm 1). The clustering iteration will be forced to stop with the maximum iteration times.

4. CLUSTERING BASED SEARCH AND ITS GPU-BASED ALGORITHM

4.1. Clustering Based Search Algorithm

The clustering based search algorithm periodically invokes the progressive clustering algorithm to assign the motion vectors of past blocks to the existing clusters, and then make incremental clustering statistics. These statistics are used as the predictors for the next blocks. Figure 10 illustrates the process of the clustering based search algorithm. The proposed search algorithm separates the blocks of a frame into multiple groups by the diagonals, and makes motion estimation group by group. For a frame consisting of $W \times H$ blocks, the blocks are separated into $W+H-1$ groups. As shown in Figure 10, the blocks with the same number belong to the same group. When one group is completed, the progressive clustering algorithm is invoked. We use the same color to represent the blocks whose motion vectors belong to the same cluster. The bottom

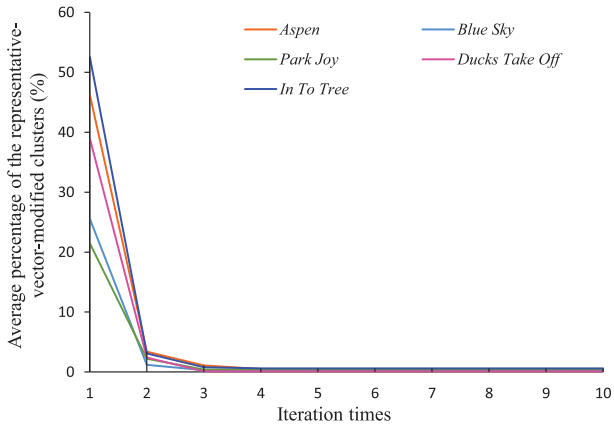


Fig. 9. The average percentage of the representative-vector-modified clusters versus iteration times.

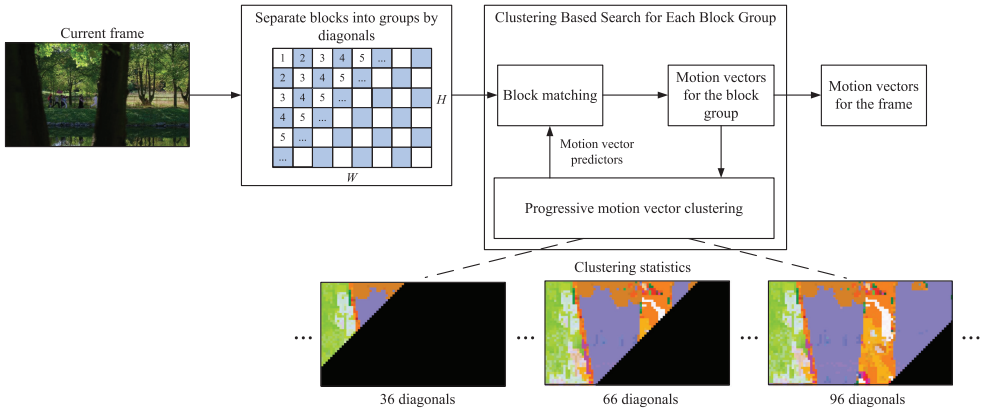


Fig. 10. The process of the clustering based search.

pictures in Figure 10 show the clustering statistics of 36, 66, and 96 block groups. With the help of the clustering statistics, some effective and efficient motion vector predictors are provided for block matching.

The block matching process integrates the spatial correlation and the clustering statistics to select motion vector predictors. The spatial correlation indicates the motion of a block has a big possibility to be close to that of its neighbors. We infer that the motion vector of a block might belong to one of the clusters which hold its neighboring blocks' motion vectors. The representative vectors of these clusters are chosen as the predictors. Figure 11 illustrates the predicted search areas and the entire search range for a block.

In Figure 11, mc_L and mc_U are the representatives of the clusters, which hold the left neighboring and the upper neighboring blocks' motion vectors respectively. mc_{max} is the representative of the cluster with the highest count. We use mc_L, mc_U and mc_{max} as the predictors to narrow the search range and reduce the candidates to be tested. Matching in the predicted search areas is probable to find the best-matching candidate, but if it fails, matching in the entire search range is performed. The matching process includes two phases.

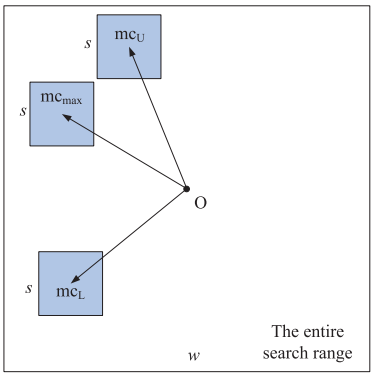


Fig. 11. The predicted search areas and the entire search range for a block.

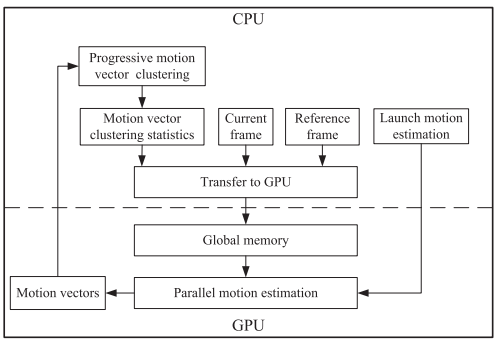


Fig. 12. High-level block diagram of the GPU-based algorithm.

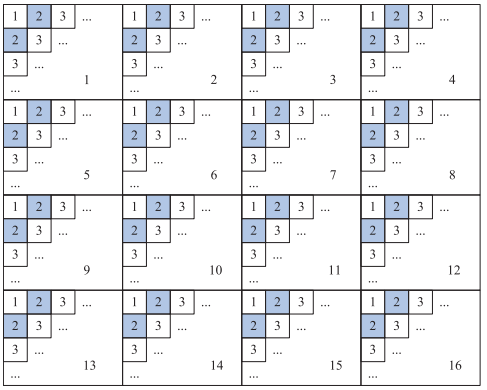


Fig. 13. Block Groups for the GPU-based algorithm.

(1) With the clustering based predictors, only the candidates in the three size s ($s \ll w$) areas are tested by the 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 will be regarded as the motion vector; otherwise, the candidates in the entire search range are tested by the line search algorithm [Huang et al. 2003] to obtain the motion vector.

We choose the line search algorithm as the strategy for the entire search range because of its effectiveness. The line search algorithm starts from searching all candidates in the three lines, and then searches additional lines in the direction of descending distortion. It stops when the MBD point is not on the boundary of searched lines.

The block matching process is applied to each block in a group. After that, the progressive clustering algorithm assigns the motion vectors of the blocks in the group to the existing clusters. Then, the clustering statistics provide predictors for the next group of blocks. We summarize the clustering based search algorithm as Algorithm 2.

4.2. GPU-Based Algorithm

The clustering based search algorithm performs motion estimation group by group. The matching process of one block is independent to those of the others in the same

ALGORITHM 2: Clustering Based Search Algorithm

Input: The blocks (B) of a frame.
Output: The motion vector of each block.
GroupByDiagonal(B);
for each block group G **do**
  for each block b in G **do**
    //The block matching process for b
    local MBD point = $SearchWithPredictors(mc_L, mc_U, mc_{max})$;
    if $Distortion$(local MBD point) $\leq T_2$ **then**
      the motion vector of $b = Displacement$(local MBD point);
    else
      the motion vector of $b = LineSearch()$;
    end
  end
  //Progressive motion vector clustering
  M = the motion vector set $\{m_1, m_2 \ldots m_w\}$ of the blocks in group G;
  E = the set of existing clusters $\{C_1, C_2 \ldots C_n\}$;
  ProgressiveClustering(M, E);
end

group. The blocks in a group can be processed in parallel. We implemented our search algorithm on a Graphics Processing Unit (GPU) with Compute Unified Device Architecture (CUDA). The high-level block diagram of the GPU-based algorithm is shown in Figure 12. The reference frame and the current frame are loaded into the global memory of GPU at the beginning of motion estimation, while the clustering statistics are periodically loaded as the predictors for block matching. The blocks in the same group are estimated parallel in GPU. Then, their motion vectors are sent to CPU to make progressive clustering statistics.

The GPU-based algorithm is hardware-dependent. In order to make better use of GPU's parallelism, the hardware architecture should be taken into account. Figure 13 takes G80 architecture with CUDA 1.0 as an example to illustrate the parallel processing of blocks in a frame. Unlike the CPU-based implementation, a video frame is divided into 16 subframes, and the blocks in each subframe are separated by the diagonals. As shown in Figure 13, the blocks with the same number belong to the same group. The number of subframes is related to the number of Stream Multiprocessors (SM) in GPU. For each time, a group of blocks are launched, and their motion vectors are estimated in parallel.

Each block is processed by a CUDA thread-block, which consists of multiple CUDA threads. The number of thread-blocks (TBN) depends on the number of the blocks in a group, that is, TBN = |The blocks in a group|. The number of threads (TN) in a thread-block is defined by $TN = max(3 \cdot s^2, w)$, where s is the size of the predicted search areas and w is the size of the entire search range. In the three predicted search areas, each candidate is tested by a CUDA thread to find the local MBD point. If the local MBD point fails in distortion check, a parallel line search algorithm is performed to find the motion vector, and the candidates in the same search line are processed by w CUDA threads in parallel. The details of the parallel matching process for a block are described in Algorithm 3, and its procedure is illustrated by an example in the online Appendix B.

5. CLUSTERING BASED AUXILIARY TRACKING AREA

The clustering statistics are also useful for predicting the position of static objects. When a target object severely changes in appearance or disappears for some time, its

ALGORITHM 3: Parallel Matching Process for a Block

Input: A block (b) and its clustering based predictors (mc_L, mc_U and mc_{max}).
Output: The motion vector of b.
local MBD point = *ParallelSearchWithPredictors*(mc_L, mc_U, mc_{max});
if *Distortion*(local MBD point) $\leq T_2$ **then**
    the motion vector of b = *Displacement*(local MBD point);
else // Parallel line search for the entire search range
    $p = 0$;
    MBD point = *ParallelSearch*(line p–1, line p, line p+1);
    if *Location*(MBD point) $\in$ line p+1 **then**
      do
        $p = p$+1;
        MBD point = *ParallelSearch*(line p+1);
      while *Location*(MBD point) $\in$ line p+1
    else if *Location*(MBD point) $\in$ line p–1 **then**
      do
        $p = p$–1;
        MBD point = *ParallelSearch*(line p–1);
      while *Location*(MBD point) $\in$ line p–1
    end
    the motion vector of b = *Displacement*(MBD point);
end

position still could be roughly estimated. Based on the clustering statistics, several auxiliary tracking areas are created to guide the object tracking.

We divide a frame into multiple blocks of the same size, and use the clustering based search for backward motion estimation. Then, depending on the motion vectors, the blocks are separated into multiple clusters. Each block is assigned to its motion vector's cluster. For each cluster, we count its member blocks that cover the target object. The cluster with the largest count is considered as the cluster of the target. An auxiliary tracking area is a block-based region, in which most of the blocks belong to the target's cluster. For simplicity, an auxiliary tracking area S is defined as a rectangle area. Suppose that S contains n blocks and wherein m ones belong to the target's cluster. S should satisfy the following criteria.

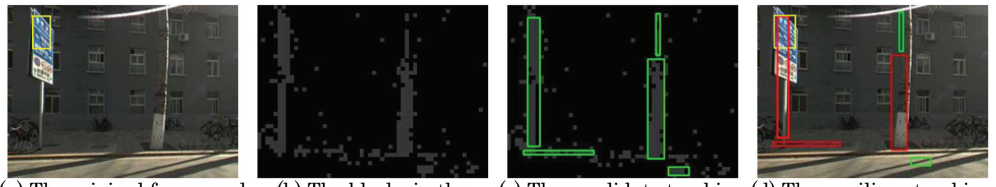
Size Criterion. The number of the blocks in S is larger than a threshold n_0.

Reliability Criterion. The ratio of the blocks which belong to the target's cluster exceeds a certain threshold p_0, that is, $m/n \geq p_0$.

The two criteria guarantee that the size of an auxiliary tracking area is relatively large, and most of its blocks belong to the target's cluster. We first find the areas which satisfy the Reliability Criterion. Then, among these areas, we choose the ones satisfying the Size Criterion as the candidate tracking areas. Let B be the set of the blocks within the target's cluster and R denote the set of candidate tracking areas. The creation of R is described as Algorithm 4.

We sort the candidate areas in R by the reliability value of m/n, and choose the first k candidates as the auxiliary tracking areas $t_1, t_2 \ldots t_k$. An example of auxiliary tracking areas selection is illustrated in Figure 14. The blocks of the original frame (Figure 14(a)) are divided into multiple clusters. The blocks in the target's cluster are shown in Figure 14(b). Then, several candidate tracking areas are created in Figure 14(c), and only three of them are selected as the auxiliary tracking areas in Figure 14(d).

For each t_i, we compute the average motion vector mv_i of its interior blocks which belongs to the target's cluster. Then, we use the average value of mv_1, $mv_2 \ldots mv_k$ to



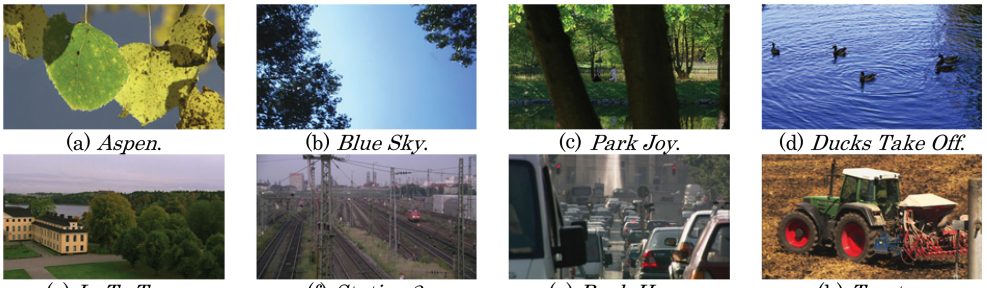
(a) The original frame and the target object. (b) The blocks in the target's cluster. (c) The candidate tracking areas. (d) The auxiliary tracking areas.

Fig. 14. The selection of auxiliary tracking areas (yellow: target object, green: candidate tracking areas, red: auxiliary tracking areas).

ALGORITHM 4: Candidate Tracking Area Creation Algorithm

Input: The blocks within the target's cluster ($B = \{b_1, b_2 \ldots b_n\}$).
Output: The set of candidate tracking areas (R).
$R = \varnothing$
// Sort B by the row-major order
SortByRow(B)
// Find the areas satisfying the Reliability Criterion
for each block b_i in B **do**
  $R' = \varnothing$ // The set of the extended candidate tracking areas
  for each area r_k in R **do**
    $r'_k = ExtendArea(r_k, b_i)$ // Extend r_k exactly enough to cover b_i
    m'_k = the total number of blocks in r'_k
    n'_k = the number of blocks belonging to the target's cluster in r'_k
    if *CheckReliability*(r'_k) **then**
      $R' = R' \cup \{r'_k\}$
    end
  end
  if $R' == \varnothing$ **then**
    $r_{new} = \{b_i\}$ // create a new candidate area
    $R = R \cup \{r_{new}\}$
  else
    // Find the area in R' with the largest value of m'_k/n'_k
    $r'_m = FindMaxReliabilityArea(R')$
    $r_m = r'_m$
    $InsertBlockToArea(b_i, r_m)$
  end
end
// Remove the areas which do not satisfy the Size Criterion
for each area r_k in R **do**
  if !*CheckSize*(r_k) **then**
    $R = R - \{r_k\}$
  end
end

estimate the position of the target in the next frame. The estimation result is applied to a predictive mean shift method [Comaniciu and Ramesh 2000] for object tracking. Because of significant appearance changes and occasional disappearances, periodic analysis of the target object and update of the target model should be taken into account. We introduce an adaptive Gaussian mixture model [McKenna et al. 1999] to boot-strap the mean shift tracker for object detection and reinitialization.



(a) *Aspen*. (b) *Blue Sky*. (c) *Park Joy*. (d) *Ducks Take Off*.

(e) *In To Tree*. (f) *Station 2*. (g) *Rush Hour*. (h) *Tractor*.

Fig. 15. The example frames of the test video sequences [Xiph.org 2013].

6. EXPERIMENT EVALUATION

6.1. Motion Estimation

To evaluate the performance of the clustering based search algorithm, we apply it to several standard high resolution video sequences from Xiph.org Video Test Media dataset [Xiph.org 2013], such as *Aspen* (1080p), *Blue Sky* (1080p), *Park Joy* (720p), *Ducks Take Off* (720p), *In To Tree* (720p), *Station 2* (1080p), *Rush Hour* (1080p) and *Tractor* (1080p). The example frames of the test video sequences are shown in Figure 15. In our experiments, the size of a block is fixed at 16×16, the search range is $(-16, 15)$, and the size of the small search areas is 3×3. As 1080 is indivisible by 16, we only take the 1920×1072 pixels of the 1080p videos into account, and the rest 8 rows of pixels are omitted. The Sum of Absolute Difference (SAD) is used as the block distortion metric. For each video sequences, we use 50 frames within it, and conduct several experiments on a computer of Intel Core2 CPU at 2.66 GHz and 2G RAM.

In the experiments, the block distortion threshold T_2 of our search algorithm is set as 2048 (see more details in the online Appendix C). We choose the mean-square error (MSE) as the criterion for measuring the performance of motion estimation algorithms. The MSE compares the motion compensated frame with the original frame. The lower the MSE, the smaller the prediction error, and therefore the more effective the motion estimation algorithm is. We compare our search algorithm with the mainstream search algorithms, including FS, UMHS [Chen et al. 2002], DS [Zhu and Ma 2000] and PLS [Huang et al. 2003], and some state-of-the-art search algorithms, such as SAAS [Shi et al. 2011] and IRS [Porto et al. 2013]. Table I shows the MSE performance for each algorithm on the test sequences. For the sequences which involve only small motions, such as *Aspen* and *Station 2*, the MSE performance of the seven algorithms are very close. On the other hand, for the sequences with large motions (such as *Blue Sky*, *Park Joy* and *Tractor*), our algorithm obviously outperforms the other fast search algorithms, only with negligibly higher MSE values than FS. This means that our clustering based search is very robust. Even when large motions are involved in the video sequences, the MSE performance of our algorithm stays very close to that of FS. The MSE values of the other fast search algorithms rise significantly for the large-motion video sequences. Especially, the MSE values of DS are nearly 2 times larger than those of ours. The MSE performance of our search algorithm is benefited from the progressive motion vector clustering, which makes incremental clustering statistics and provides efficient predictors for block matching. With the help of these predictors, most of the blocks could find their best-matching candidates.

We test the time cost per frame for each algorithm on the test sequences, and Table II shows the results. The time cost of our clustering based search algorithm is the sum of the block matching time and the clustering time. Compared to FS, our algorithm only

Table I. MSE Performance Comparison

Video Sequence	Search Algorithm						
	Ours	FS	UMHS	DS	PLS	SAAS	IRS
Aspen	21.1	18.24	21.18	23.15	19.15	21.7	20.81
Blue Sky	32.16	26.46	51.27	77.22	33.36	53.61	54.71
Park Joy	283.58	269.8	336.31	562.57	344.39	342.9	428.29
Ducks Take Off	102.83	102	102.31	134.33	103.56	106.7	123.05
In To Tree	35.64	31.39	32.79	84.14	32.13	35.32	45.82
Station 2	12.67	11.42	15.87	22.28	13.9	17.82	16.75
Rush Hour	32.1	31.41	35.64	51.63	33.78	38.51	43.62
Tractor	62.27	58.41	66.94	103.7	64.27	70.12	82.14
Average	72.79	68.64	82.78	132.37	80.56	85.84	101.89

Table II. Comparison of the Time Cost per Frame (ms/frame)

Video Sequence	Search Algorithm						
	Ours	FS	UMHS	DS	PLS	SAAS	IRS
Aspen	77.78	3283	368.67	75.86	399.76	165.6	139.79
Blue Sky	77.87	3255	355.45	116.55	381.35	157.2	185.6
Park Joy	89.96	1460	171.67	54.12	183.34	75.4	106.3
Ducks Take Off	47.49	1440	127.78	25.86	139.39	57.5	48.61
In To Tree	29.89	1445	145.65	28.39	152.86	64.8	53.13
Station 2	74.93	3236	439.83	98.38	364.52	152.6	133.42
Rush Hour	104.36	3245	434.91	85.25	432.41	172.45	170.21
Tractor	121.03	3238	390.36	129.81	403.46	191.68	208.05
Average	77.91	2575.3	304.29	76.77	307.14	129.65	130.63

takes 3% of its time cost to obtain approximate MSE values. On average, the speedup ratio of our algorithm is nearly 33 times. Our algorithm also shows a faster speed than UMHS, PLS, SAAS and IRS. Specifically, it performs 4 times faster than the UMHS and PLS algorithms, and 1.6 times faster than SAAS and IRS. The DS algorithm makes a tradeoff between the speed and the MSE performance. Although DS is a little faster than our algorithm on average, the MSE values of DS are much higher, especially for the large-motion video sequences.

For further complexity evaluation, we calculate the searched candidates per block for the test video sequences. Table III shows the comparison results between our search algorithm and the other algorithms. The average number of the searched candidates by our algorithm is 28.7. Compared with the FS algorithm, which searches the 1024 candidates in the search range, our algorithm saves 97.2% search overhead but achieves very close MSE performance. On average, the candidates searched by our algorithm are much fewer than those by UMHS (107.5), PLS (116.3), SAAS (47.1) and IRS (47.7). Only the DS algorithm searches slightly fewer candidates (28.6) than ours, but it causes a serious MSE rise. UMHS, PLS and SAAS straightforwardly apply the motion vectors of neighboring blocks as the predictors. Compared with them, the effectiveness of applying clustering statistics could be clearly seen. The candidates searched by our algorithm are significantly reduced by the clustering based predictors. For most of the blocks, our algorithm only needs to test the candidates in the three predicted search areas to find their motion vectors.

From these experiment results, it is clear that, our search algorithm has the capability to significantly reduce the searched candidates, only with a negligible increase in MSE. The performance of our search algorithm is attributed to the predictors provided by the clustering statistics (see more comparison results with K-means [MacQueen

Table III. Comparison of the Searched Candidates per Block

Video Sequence	Search Algorithm						
	Ours	FS	UMHS	DS	PLS	SAAS	IRS
Aspen	21.1	1024	107.5	21	123.2	48.3	38.7
Blue Sky	21.3	1024	103.8	35	117.4	45.9	54.1
Park Joy	59.6	1024	116.5	40.7	126.1	51.2	73.4
Ducks Take Off	31.2	1024	85.6	24.3	96	38.5	31.2
In To Tree	18.7	1024	94	20	104.6	41.8	33.6
Station 2	18.2	1024	122.1	27.6	110.3	45.2	38.3
Rush Hour	27.1	1024	122.5	23.8	131.1	50.3	50.2
Tractor	32.4	1024	108.1	36.5	121.3	56.2	62.1
Average	28.7	1024	107.5	28.6	116.3	47.1	47.7

Table IV. Time Cost of Clustering per Frame versus Time Cost of Total Motion Estimation (ms/frame)

Video Sequence	Time Cost		
	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%
Station 2	74.93	3.7	4.94%
Rush Hour	104.36	4.8	4.60%
Tractor	121.03	6.5	5.37%
Average	77.91	3.71	4.70%

1967] and DBSCAN [Ester et al. 1996] in the online Appendix C). In order to measure the overhead of our progressive clustering, we test the time cost of clustering per frame. Table IV shows the time cost of clustering and the time cost of total motion estimation. The time cost of clustering only measures the clustering time per frame, and the time cost of total motion estimation is the sum of the block matching time and the clustering time. As shown in Table IV, the time cost of clustering only occupies 4.7% of total motion estimation on average. The overhead of clustering is very low, but the motion estimation performance is significantly improved by the clustering statistics.

We also calculate the blocks which are only searched with the clustering based predictors, and the other blocks which are searched by the line search. As Table V shows, for the majority of the test video sequences, more than 90% of the blocks benefit from the clustering statistics, and they are only searched in the three predicted search areas to obtain the motion vectors. Especially for *Aspen*, *Blue Sky*, *In To Tree* and *Station 2*, only less than 3% of the blocks use the line search. However, the percentage of the blocks searched by the line search obviously rises for *Ducks Take Off* (13.1%) and *Park Joy* (28.3%). The main reason is that *Ducks Take Off* involves severe deformations and *Park Joy* involves heavy occlusions. For some blocks, severe deformations and heavy occlusions may affect the displacements of their best-matching candidates, and their motion vectors may not reflect the actual movement of the scene. In such cases, the clustering information of these blocks could not provide appropriate predictors for their neighbors, and the percentage of the blocks searched by the line search is increased. Although, the usage of the line search rises for *Ducks Take Off* and *Park Joy*, our search algorithm still shows superior performance to the other algorithms in terms of MSE and search speed.

Table V. Blocks Only Searched with the Clustering-Based Predictors versus Blocks Searched by the Line Search

Video Sequence	Total blocks per frame	Blocks only searched with the clustering based predictors		Blocks searched by the line search	
		Count	Percentage	Count	Percentage
Aspen	8040	7894	98.1%	146	1.9%
Blue Sky	8040	7867	97.8%	173	2.2%
Park Joy	3600	2583	71.7%	1017	28.3%
Ducks Take Off	3600	3130	86.9%	470	13.1%
In To Tree	3600	3566	99.0%	34	1.0%
Station 2	8040	8004	99.5%	36	0.5%
Rush Hour	8040	7772	96.6%	268	3.4%
Tractor	8040	7325	91.1%	715	8.9%
Average	6375	6018	94.4%	357	5.6%

Table VI. Comparison of the Performance between CPU and GPU

Video Sequence	CPU		GPU		
	MSE[a]	TC[b]	MSE[a]	TC[b]	Speedup
Aspen	21.1	77.78	20.5	20.1	3.9
Blue Sky	32.16	77.87	32.64	18.9	4.1
Park Joy	283.58	89.96	285.42	29.2	3.1
Ducks Take Off	102.83	47.49	102.57	11.3	4.2
In To Tree	35.64	29.89	33.96	7.6	3.9
Station 2	12.67	74.93	13.2	19.1	3.9
Rush Hour	32.1	104.36	31.76	25.4	4.1
Tractor	62.27	121.03	60.83	31.8	3.8
Average	72.79	77.91	72.61	20.4	3.8

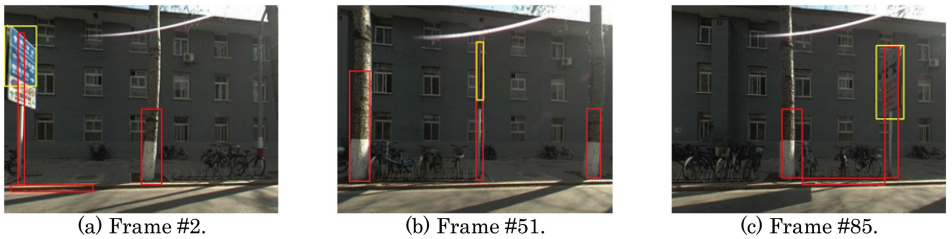
[a]Mean-Square Error. [b]Time Cost per frame (ms/frame).

Furthermore, we implement our search algorithm on a NVIDIA 8800GTX graphics card with CUDA. CPU makes progressive clustering statistics, while GPU executes parallel block motion estimation. As shown in Table VI, the experiment results in the GPU implementation get around 4 times faster. Table VI also shows a slight difference between the MSE performance of the CPU and GPU implementations. The main reason is that some of the blocks use different predictors. We use different ways to group the blocks in the CPU and GPU implementations. As a result, the clustering statistics in the two implementations are different.

6.2. Persistent Static Object Tracking

In order to evaluate the effectiveness of auxiliary tracking areas, we capture several videos by a vehicle-mounted camera on the roads in our campus, and choose the road signs in the videos as the target objects. The resolution of each video is 640×480. The video frames are divided into size 16×16 blocks for motion estimation, and the parameters for the auxiliary tracking areas are set as follows: $n_0 = 8$, $p_0 = 0.8$ and $k = \sim 3$. The tracking experiments are performed on a computer of Intel Core2 CPU at 2.66 GHz and 2G RAM.

Figure 16 shows our tracking results and the auxiliary tracking areas created by the clustering statistics. The target object is a road sign with different appearances on the two sides. As the camera moves, the target object almost disappears in some frames. Our method uses the auxiliary tracking areas to estimate the position of the road sign.



(a) Frame #2. (b) Frame #51. (c) Frame #85.

Fig. 16. Our tracking results for Road Sign #1 (red: auxiliary tracking areas, yellow: tracking results).

Even when the appearance of the road sign changes severely or it almost disappears, the road sign still could be tracked.

We compare our tracking method with other on-line tracking methods, such as the original mean-shift tracking [Comaniciu et al. 2000], the adaptive color-based particle filter [Nummiaro et al. 2003], the on-line boosting tracking [Grabner and Bischof 2006] and our previous work [Zhou et al. 2012]. We test these tracking methods on three road sign video sequences. Figure 17 shows the tracking results of the different methods (see more results in the online Appendix D).

As Figure 17 shows, the original mean-shift tracking, the adaptive color-based particle filter and the on-line boosting tracking fail to track and lose the target road sign. Due to camera movement, the appearance of the road sign changes significantly. The color histograms and the color features on the two sides of the road signs are quite different. Moreover, the road sign almost disappears in some frames. The pixels of the road sign are insufficient to provide valid color histograms and features. It is very difficult for them to track the road sign persistently. Compared with our previous work, the tracking results with the clustering based auxiliary tracking areas are shown to be more accurate in terms of target position. This is because the clustering based auxiliary tracking areas could provide a more precise estimation of the road sign position.

For further comparison with our previous work, we test the time cost of tracking for different auxiliary tracking area creation approaches. As shown in Table VII, the time cost of the auxiliary tracking area creation has a minor increase, when the clustering statistics are used. On average, the creation time rises by 4.76 ms/frame. The clustering based auxiliary tracking areas estimate the target position more accurately, and the time cost of the mean-shift tracking is reduced. In general, the clustering statistics improve the target position estimation only with a negligible tracking time increase.

7. CONCLUSION

Besides the spatial correlation of motion vectors, those nonneighboring blocks may also have similar motions due to close depths or belonging to the same object inside the scene. Therefore, the motion vectors usually have several patterns, and these patterns indicate that the motion vectors have a clustering structure. This article gives a systematic description of motion vector clustering and proposes a progressive clustering algorithm. The clustering algorithm periodically counts the motion vectors of the past blocks to make incremental clustering statistics.

The clustering statistics are used as the motion vector predictors. It is proved to be much more efficient for one block to find the best-matching candidate with these predictors. From the experiment results, we can see that the MSE performance of our clustering based search algorithm is very close to that of the FS algorithm while the speed of our algorithm is 33 times faster on average. It is also shown that our algorithm

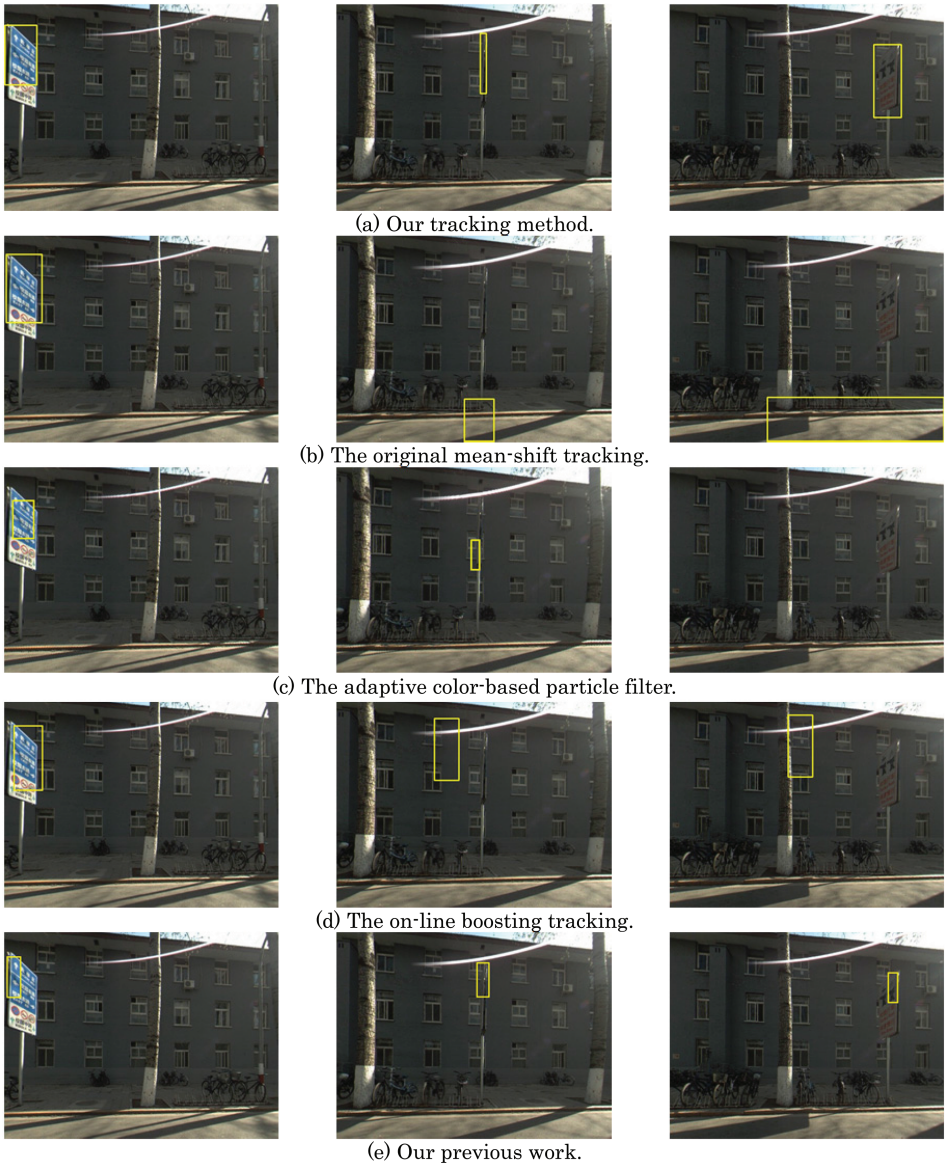


Fig. 17. The tracking results of Road Sign #1 with different tracking methods. The first, second, and third columns show frame #2, frame #51, and frame #85, respectively.

outperforms the UMHS, DS, PLS, SAAS and IRS algorithms, especially for the video sequences with large motions. We design the GPU-based algorithm for our clustering based search which achieves nearly 4 times speedup.

The clustering statistics are also applied in persistent static object tracking. Based on them, several auxiliary tracking areas are created to guide the object tracking. Even when the target object has significant changes in appearance or occasionally disappears, its position still can be estimated by the auxiliary tracking areas. The

Table VII. Time Cost of Tracking for Different Auxiliary Tracking Area Creation Approaches (ms/frame)

	The average motion vector approach		
Video Sequence	Auxiliary tracking areas creation	Mean-shift tracking	Total
Road Sign #1	23.2	12.9	36.1
Road Sign #2	29.8	13.5	43.3
Road Sign #3	28.5	12.7	41.2
Average	27.17	13.03	40.2
	The clustering statistics approach		
Video Sequence	Auxiliary tracking areas creation	Mean-shift tracking	Total
Road Sign #1	27.8	9.4	37.2
Road Sign #2	35.1	10.3	45.4
Road Sign #3	32.9	9.8	42.7
Average	31.93	9.83	41.77

tracking experiments demonstrate that the auxiliary tracking areas help to locate the target object effectively.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

REFERENCES

S. Avidan. 2004. Support vector tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* 26, 8, 1064–1072.

K. Chen, Z. Zhou, and W. Wu. 2012. Clustering based search algorithm for motion estimation. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME'12)*. IEEE, 622–627.

Z. Chen, P. Zhou, and Y. He. 2002. Fast integer pel and fractional pel motion estimation for JVT. In *Proceedings of the 6th Meeting of JVT-F017. Joint Video Team of ISO/IEC MPEG & ITU-T VCEG*. 5–13.

D. Comaniciu and V. Ramesh. 2000. Mean shift and optimal prediction for efficient object tracking. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'00)*. IEEE, 70–73.

D. Comaniciu, V. Ramesh, and P. Meer. 2000. Real-time tracking of non-rigid objects using mean shift. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00)*. IEEE, 2, 142–149.

R. Cucchiara, A. Prati, and R. Vezzani. 2003. Object segmentation in videos from moving camera with MRFs on color and motion features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*. IEEE, 405–410.

D. L. Davies and D. W. Bouldin. 1979. A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell.* 1, 2, 224–227.

W. Du and J. Piater. 2008. A probabilistic approach to integrating multiple cues in visual tracking. In *Proceedings of the 10th European Conference on Computer Vision (ECCV'08)*. Springer, 225–238.

M. Ester, H. P. Kriegel, J. Sander, and X. Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*. AAAI, 96, 226–231.

M. Gelgon and P. Bouthemy. 2000. A region-level motion-based graph representation and labeling for tracking a spatial image partition. *Pattern Recognit.* 33, 4, 725–740.

H. Grabner and H. Bischof. 2006. On-line boosting and vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06)*. IEEE, 260–267.

B. Han, S. W. Joo, and L. S. Davis. 2007. Probabilistic fusion tracking using mixture kernel-based Bayesian filtering. In *Proceedings of the 11th International Conference on Computer Vision (ICCV'07)*. IEEE, 1–8.

Y. W. Huang, S. Y. Ma, C. F. Shen, and L. G. Chen. 2003. Predictive Line Search: an efficient motion estimation algorithm for MPEG-4 encoding systems on multimedia processors. *IEEE Trans. Circ. Syst. Video Tech.* 13, 1, 111–117.

C. Hennebert, V. Rebuffel, and P. Bouthemy. 1996. In *Proceedings of the 13th IEEE International Conference on Pattern Recognition (ICPR'96)*. IEEE, 218–222.

T. Koga, K. Linuma, A. Hirano, Y. Iijima, and T. Ishiguro. 1981. Motion-compensated interframe coding for video conferencing. In *Proceedings of the National Telecommunication Conference*. IEEE, G5.3.1–G5.3.5.

E. Krause. 1987. *Taxicab Geometry: An Adventure in Non-Euclidean Geometry*. Dover Publications, New York.

J. Kwon and K. M. Lee. 2010. Visual tracking decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'10)*. IEEE, 1269–1276.

R. Li, B. Zeng, and M. L. Liou. 1994. A new three-step search algorithm for block motion estimation. *IEEE Trans. Circ. Syst. Video Technol.* 4, 4, 438–442.

J. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press, 1, 14, 281–297.

S. J. McKenna, Y. Raja, and S. Gong. 1999. Tracking color objects using adaptive mixture models. *Image Vision Comput.* 17, 3, 225–231.

K. Nummiaro, E. Koller-Meier, and L. Van Gool. 2003. An adaptive color-based particle filter. *Image Vision Comput.* 21, 1, 99–110.

K. Okuma, A. Taleghani, N. De Freitas, J. J. Little, and D. G. Lowe. 2004. A boosted particle filter: Multitarget detection and tracking. In *Proceedings of the 8th European Conference on Computer Vision (ECCV'04)*. Springer, 28–39.

L. M. Po and W. C. Ma. 1996. A novel four-step search algorithm for fast block motion estimation. *IEEE Trans. Circ. Syst. Video Technol.* 6, 3, 313–317.

M. Porto, C. Cristani, P. Dall'Oglio, M. Grellert, J. Mattos, S. Bampi, and L. Agostini. 2013. Iterative random search: a new local minima resistant algorithm for motion estimation in high-definition videos. *Multimedia Tools Appl.* 63, 1, 107–127.

D. A. Ross, J. Lim, R. S. Lin, and M. H. Yang. 2008. Incremental learning for robust visual tracking. *Int. J. Computer Vision* 77, 1–3, 125–141.

Z. Shi, W. A. C. Fernando, and D. V. S. De Silva. 2010. A motion estimation algorithm based on predictive intensive direction search for H. 264/AVC. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'10)*. IEEE, 667–672.

Z. Shi, W. A. C. Fernando, and A. Kondoz. 2011. Adaptive direction search algorithms based on motion correlation for block motion estimation. *IEEE Trans. Consum. Electron.* 57, 3, 1354–1361.

H. Tao, H. S. Sawhney, and R. Kumar. 2002. Object tracking with Bayesian estimation of dynamic layer representations. *IEEE Trans. Pattern Anal. Mach. Intell.* 24, 1, 75–89.

Y. W. Wu, B. Ma, and P. Li. 2012. A variational method for contour tracking via covariance matching. *Science China Info. Sci.* 55, 11, 2611–2623.

Xiph.org. 2013. Xiph.org video test media (derf's collection). http://media.xiph.org/video/derf/.

A. Yilmaz, O. Javed, and M. Shah. 2006. Object tracking: A survey. *ACM Comput. Surv.* 38, 4, 13.

Z. Yin and R. T. Collins. 2008. Object tracking and detection after occlusion via numerical hybrid local and global mode-seeking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*. IEEE, 1–8.

Y. Zhou, Z. Zhou, K. Chen, and W. Wu. 2012. Persistent object tracking in road panoramic videos. In *Proceedings of the 13th Pacific Rim Conference on Multimedia (PCM'12)*. Springer, 359–368.

C. Zhu, X. Lin, L. P. Chau, K. P. Lim, H. A. Ang, and C. Y. Ong. 2001. A novel hexagon-based search algorithm for fast block motion estimation. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP'01)*. IEEE, 1593–1596.

S. Zhu and K. K. Ma. 2000. A new diamond search algorithm for fast block-matching motion estimation. *IEEE Trans. Image Process.* 9, 2, 287–290.

Received March 2014; revised July 2014, September 2014; accepted September 2014