Computer Vision and Image Understanding 170 (2018) 14–27

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

PA-Search: Predicting units adaptive motion search for surveillance video coding

Yonghong Tian*[,a], Jiaying Yan[b], Siwei Dong[a], Tiejun Huang[a]

[a] National Engineering Laboratory for Video Technology, School of Electronics Engineering and Computer Sciences, Peking University, Rm. 2608, Sci. Bldg. 2, Peking University, 5 Yiheyuan Rd., Beijing 100871, China

[b] School of Electronic and Computer Engineering, Shenzhen Graduate School, Peking University, Rm. 2604, Sci. Bldg. 2, Peking University, 5 Yiheyuan Rd., Beijing 100871, China

ARTICLE INFO

Keywords:
Surveillance video coding
Motion search
Predicting unit classification
PA-Search
HEVC

MSC:
41A05
41A10
65D05
65D17

ABSTRACT

The large scale of surveillance video and the high requirement of compression in time requires a low complexity and high efficiency compression algorithm to compress surveillance video. Motion search is a very time-consuming procedure in video coding. In the recent video coding standards such as HEVC/H.265, this procedure becomes more flexible by utilizing the division structure of Coding Units (CUs) and Predicting Units (PUs). However, for surveillance videos that are often captured by fixed-view cameras, the used motion search strategy still does not make full use of their intrinsic characteristics. To address this problem, we propose a PU-Adaptive Search (PA-Search) method for surveillance videos. In PA-Search, a background model is firstly constructed for a super group of pictures and then a background-foreground representation (BFR) is derived for each frame in this group. Utilizing the BFR, PUs are classified into four categories, namely, Full Background PUs (FBPUs), Background PUs (BPUs), Foreground PUs (FPUs), and hybrid foreground-background PUs (XPUs). In PA-Search, zero motion vector (zero-MV) and non-sub-pixel search are assigned to FBPUs and an error-tolerant search algorithm is also performed to reduce the influence of PU mis-classifications; while for non-FBPUs, adaptive search range is calculated according to the PU category and its size, and a BFR-based early-termination algorithm is also used to reduce the search complexity. Moreover, an early terminate partition algorithm is adopted by Full Background CUs to further reduce the encoding time. Experimental results demonstrate the advantage of the proposed PA-Search on HEVC reference software HM-16.0. PA-Search can reduce the number of search points and the total encoding time averagely by 66.90% and 46.69% over TZ Search, while maintaining the coding efficiency.

1. Introduction

With the wide application of the surveillance cameras in social safety, city traffic management, and home care, great challenges are brought to the high efficient video compression. For example, about 5 million surveillance cameras were deployed in UK in 2012. If these cameras were all High-Definition (HD) ones and the generic video codecs such as H.264/AVC were adopted to compress the videos, hundreds of Terabytes data would be produced per minute or thousands of Petabytes per month (Zhang et al., 2014b). Thus to realize real-time security monitoring as well as long-time archiving, there is a great demand for high-efficiency and low-complexity surveillance video coding methods. Based on these requirements for surveillance video coding, this paper focuses on the research on the low-complexity surveillance video coding.

In video coding, motion estimation (ME) often plays an important role in reducing the temporal redundancy. To achieve better estimation performance, motion search is thus conducted for each to-be-encoded block in the current frame (called the *current block* hereafter) to find the best-matched block in the reference frames so as to provide precise prediction. Nevertheless, motion search is also a very time-consuming procedure. In the HEVC reference software HM (Bossen et al., 2012), motion search takes 7.4% of the total processing time. The proportion is even much larger in hardware encoders. For example, it has been reported that one third of processor cycles and 90% of the total memory access are dedicated to motion search and estimation in the hardware codec (Lou et al., 2010). Therefore, fast motion search methods are highly desired to the effective implementation of video encoders.

Typically, for each current block, the computational complexity of motion search can attribute to two factors: the number of the candidate

* Corresponding author.
E-mail address: yhtian@pku.edu.cn (Y. Tian).

https://doi.org/10.1016/j.cviu.2018.02.009
Received 16 May 2017; Received in revised form 19 December 2017; Accepted 27 February 2018
Available online 10 March 2018
1077-3142/

blocks (called *search points* hereafter), and the measure of block matching. Obviously, the full/exhaustive search over the entire search window in the reference frames gives the optimal match; however, this is impractical due to time complexity. That is, search should be performed only at a few selected locations within the search range guided by some fast search strategies. During the search process, there are many choices for block matching measure, such as mean-square-error (MSE) and sum of absolute difference (SAD). With the same size of blocks, SAD is more appealing to video coding for its simplicity and performance (Dhara et al., 2010). Thus given a fixed procedure of the calculation of SAD between two blocks, how to effectively reduce the search points is crucial to speed up the motion search process.

In general, three fast search approaches have been investigated in the literature. The first one is to design the fast search pattern that utilizes some selected points to find out the matched block, such as octagonal search (Dhara et al., 2010), diamond search (Zhu and Ma, 2000), UMHexagon Search (Chen et al., 2003), and TZ Search (JVT of ISO/IEC MPEG, ITU-T VCEG, 2010). An underlying assumption is that the error surface is unimodal, i.e., the block distortion error decreases monotonically as the search point moves closer to the global minimum (Dhara et al., 2010). Instead of utilizing a pre-defined search pattern that has to be isotropic with respect to the current point, the second approach is to adopt the adaptive search range (ASR) strategy that adjusts the search ranges so as to dynamically reduce the search points. For example, the ASR of the current block was determined by the motion vector (MV) of its father macroblock in Chen et al. (2007), by the variance of a motion vector predictor (MVP) set in Chung-Cheng Lou and Kuo (2010); Lou et al. (2010), or by prediction error and local statistics of the neighboring blocks in Paul et al. (2008). Different from the above two approaches, the third one aims to early terminate the motion search. By utilizing some thresholds based on statistical values regarding the current block and previously-coded blocks to determine whether to early terminate the search, some methods (e.g., Sarwer and Wu, 2009; Yang et al., 2002; Yang et al., 2005) could effectively reduce the computation of ME.

As a new-generation video coding standard, HEVC/H.265 (Bross et al. (2012) leads the video coding performance to a new milestone. Instead of utilizing fixed-size coding blocks (and macroblocks), HEVC introduces a quad-tree division structure to represent variable-size coding blocks (called coding units, CUs). In quad-tree coding, the largest CU size is often set to 64 × 64 (i.e., the CU depth is 0) while the smallest is 8 × 8 (i.e., the CU depth is 3). Each CU can be then divided into prediction units (PUs) of either intra-picture or inter-picture prediction type which can vary in size from 64 × 64 to 4 × 4. This quad-tree CU division structure makes the motion search more precise, more flexible and also more time-consuming. By balancing the search complexity and rate-distortion performance, TZ Search is adopted by HEVC. In Pan et al. (2013), an Early-Termination TZ Search algorithm (ETTZ) was proposed by searching a region around a MVP to test whether the MVP is precise enough to skip the search, consequently saving search points remarkably. In Shen et al. (2014), a fast inter-mode decision algorithm for HEVC by jointly using the inter-level correlation of quadtree structure and the spatiotemporal correlation was proposed to reduce the computational comlexity.

Despite demonstrating the promising speed-up performance, these fast motion search methods, however, are not specifically designed for surveillance videos that are often captured by fixed-view cameras. In the surveillance scene, there always exist some static background regions. Thus the motion search can be significantly simplified in these regions. Following this idea, (Ma et al., 2015; Wang and Dong, 2014; Xing et al., 2013; Zhang et al., 2013) proposed some methods to reduce the complexity of the encoder. Our previous work Zhao et al. (2014) proposed a background-foreground-division-based search (BFDS) method. The experimental results show that compared with TZ Search, BFDS could significantly reduce the number of search points on surveillance videos. However, when the foreground regions are relatively large, BFDS does not perform well since the complex search strategy in these regions will increase the total search complexity. Besides, BFDS doesn't make full use of background-foreground information in CU division and sub-pixel motion estimation, so its time saving is relatively small.

To address this problem, we propose a PU-Adaptive Search (PA-Search) method for surveillance videos. In PA-Search, a background model is firstly constructed for a super-group of pictures (SGOP) where each SGOP often consists of several GOPs and then a background-foreground representation (BFR) is derived for each frame in this group. Utilizing the BFR, PUs are classified into four categories, namely, Full Background PUs (FBPUs), Background PUs (BPUs), Foreground PUs (FPUs) and hybrid foreground-background PUs (XPUs). After that, zero-MV and non-sub-pixel search are assigned to FBPUs; while for non-FBPUs, adaptive search range is calculated according to the PU category and its size. That is, a larger search range is adopted for XPUs and a smaller one for BPUs, while that for FPUs is between the two categories. In the same category, PUs with smaller size always have a larger search range. Meanwhile, an error-tolerant search algorithm is also performed to reduce the influence of PU mis-classifications (i.e., wrongly classifying a BPU, FPU or XPU into 'FBPU'), and a BFR-based early-termination search algorithm is used to determine whether to early terminate the search procedure on a non-FBPU. Moreover, an early terminate partition strategy is adopted by Full Background CUs to further reduce encoding time. As such, PA-Search can significantly reduce both the search points and the total encoding time while maintaining the coding efficiency.

Extensive experiments were conducted on sixteen surveillance videos from the PKU-SVD-A dataset. The experiments were performed on the recent version of HEVC reference software, HM-16.0, where the original TZ Search was selected as the anchor. Moreover, BFDS (Zhao et al., 2014) and Ma et al. (2015) were also used for comparison. The experimental results show that PA-Search can significantly reduce search points and total encoding time on HM-16.0 (averagely 66.90% and 46.69% over TZ Search) while maintaining the coding efficiency with only 0.70% BD-rate loss negligibly.

The rest of this paper is organized as follows: Section 2 describes the related work. Section 3 analyzes the possibility of adopting different search strategies for foreground and background regions. The proposed PA-Search is presented in Section 4. Experimental results are shown in Section 5. Finally, this paper is concluded in Section 6.

2. Related work

As shown in Fig. 1, for each block (e.g., a macroblock, a CU or a PU), the motion search task is to find a block that matches best in the reference frames (which are already encoded), where the best match means that the block can minimize an error measure (e.g., SAD) within the search range. Formally, a block with size of S_b is denoted as $b = I(x, y, S_b)$, where (x, y) is the left-top position of the block in the current frame I. Let $F(x + u, y + v, S_b)$ be the best-matched block in the reference frame F where the MV is defined as (u, v), then the motion search process can be expressed as a rate-distortion (RD) motion

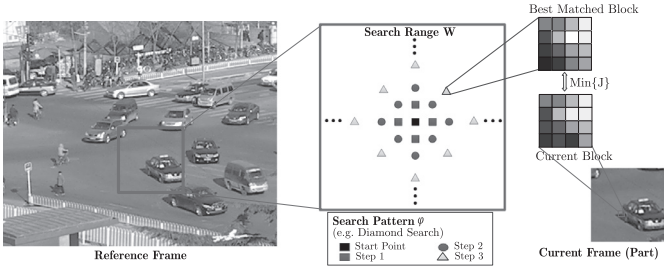


Fig. 1. Schematic diagram of motion search.

estimation problem (Girod, 1994) as (1).

$$J_{\lambda}^{(W,\varphi)}(b) = \min_{\substack{(u,v)\le W \\ (u,v)\leftarrow\varphi(x,y)}} \begin{bmatrix} \mathscr{D}(I(x,y,S_b) \\ -F(x+u,y+v,S_b)) \\ +\lambda\mathscr{R}(u,v) \end{bmatrix} \tag{1}$$

where W is the search range that can be expressed as the size of a search window around the location (x, y), $\varphi(x, y)$ is a search pattern that calculates the candidate blocks (*a.k.a.* search points) with respect to the current block b following some heuristic strategies (it can also be represented as an order set of the candidate blocks), $\mathscr{D}(\cdot)$ denotes the block matching measure while $\mathscr{R}(u, v)$ denotes the MV bitrate, and λ is the Lagrangian multiplier. Obviously, the computation involved in one search operation and the number of search points are two major factors to determine the search complexity. The cost of each search operation can be made smaller by either accelerating the SAD calculation (Lin and Tai, 1997) or by performing subsampling or partial Lagrangian computation techniques (Kossentini et al., 1997). However, it is impractical to conduct the full search over the entire search window in the reference frames. That is, search should be performed only at a few selected locations within the search range guided by some fast search strategy. Thus, the search optimization can be casted as a constrained problem that is *to find an optimal search range and search pattern* (W_b^*, φ_b^*) *for a block* $b = I(x, y, S_b)$ *that can meet the RD cost criterion using the minimal search complexity, which can be expressed as (2).*

$$\begin{aligned} &(W_b^*, \varphi_b^*) = \underset{W,\varphi}{argmin}\,\mathscr{L}_b(W, \varphi) \\ &s.\,t.\;\; W \le W_{Max},\ \varphi \subseteq \Phi,\ \text{ and } J_{\lambda}^{(W,\varphi)}(b) \le T_{RD} \end{aligned} \tag{2}$$

where $\mathscr{L}_b(W, \varphi)$ denotes the search complexity given a search range W and a search pattern φ for the block b, W_{Max} denotes the maximal search range, Φ is the pre-defined set of search patterns, while T_{RD} is a pre-defined threshold for motion-compensated coding efficiency.

Along with this optimization problem, various fast search methods have been developed to reduce the search complexity, which can be roughly divided into three categories: 1) To dynamically calculate the adaptive search range (ASR) W; 2) To design or select the fast search pattern $\varphi^* \subseteq \Phi$; and 3) To early terminate the search procedure. Thus this section will present a brief review of their related works, and then discuss how to optimize the CU mode decision in HEVC and how to conduct effective motion search on surveillance videos.

2.1. Adaptive search range prediction

Traditionally, the search range can be pre-defined or selected from a list of candidates or the MVs previously obtained for adjacent blocks (Paul et al., 2008). However, the pre-defined candidates are difficult to reflect the large variations in actual motion. Instead, several ASR prediction methods have been investigated recently. For example, Chen *et al.* proposed a macroblock-level ASR algorithm in Chen et al. (2007), where the search range of the current block was determined by the MV of its father macroblock; while in Paul et al. (2008), the search range was determined by prediction error and local statistics of the neighboring blocks.

Specifically, Chung-Cheng Lou and Kuo (2010); Lou et al. (2010) proposed an ASR prediction algorithm by utilizing the variance of a motion vector predictor (MVP) set. In this algorithm, a joint MVP set N_{st} is defined for each block, including a spatial MVP set $N_s = \{mv_a, mv_b, mv_c, mv_d\}$ that contains four upper-left neighboring MVs and a temporal one $N_t = \{mv_0 \sim mv_8\}$ that contains nine MVs of the previous frame. Then the best MVP, MVP_{bst}, is calculated as the mean value of N_{st} and can be used as the center of motion search. The variance, σ_{st}, which reflects the consistency of the MVP set, can be used to calculate the search range. That is, a larger variance implies lower accuracy of the MVP set and, thus, a larger search range. However, the MVP_{bst} calculated by the MVs of the previously-encoded blocks cannot work well if the current block contains a fast-moving object. As a result, this algorithm cannot achieve the best performance.

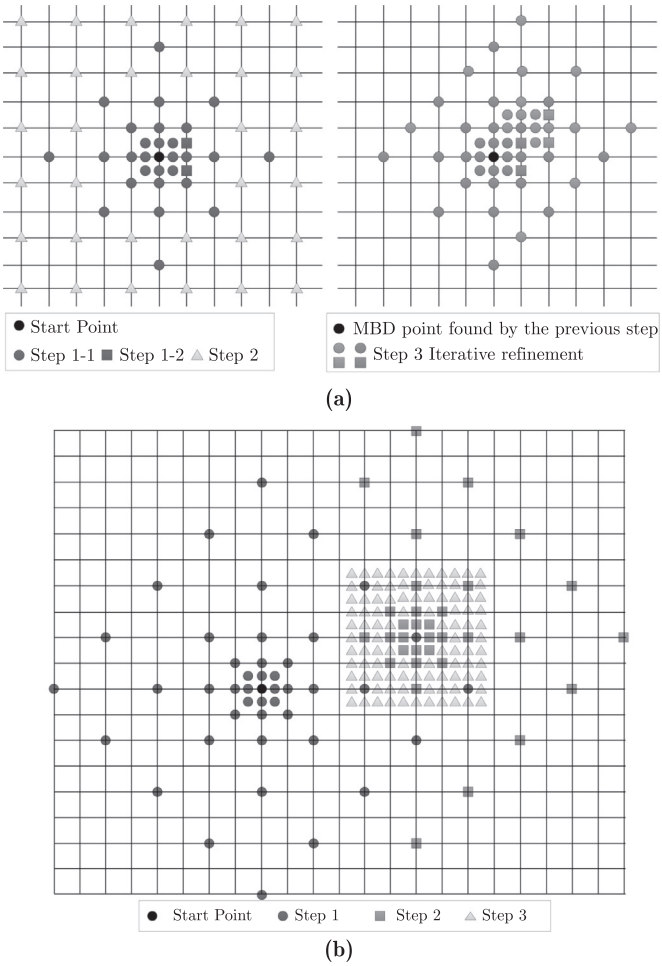


Fig. 2. (a) Search patterns in TZ Search (JVT of ISO/IEC MPEG, ITU-T VCEG, 2010); (b) The modified TZ search pattern for foreground regions in BFDS (Zhao et al., 2014).

2.2. Fast search patterns

In order to only utilize some selected points to find out the matched block, various fast search patterns were proposed in recent years, such as octagonal search (Dhara et al., 2010), diamond search (Zhu and Ma, 2000), UMHexagon Search (Chen et al., 2003), and TZ Search (JVT of ISO/IEC MPEG, ITU-T VCEG, 2010). Among them, TZ Search is adapted by HEVC reference software HM, while UMHexagon Search is used in H.264 reference software JM.

Basically, TZ Search utilizes the multiple MVPs decision to locate an initial search point and hybrid block-matching search to find the best-matched block (JVT of ISO/IEC MPEG, ITU-T VCEG, 2010). As shown in Fig. 2(a), after the starting MVP point is determined, TZ Search combines multiple diamond/square search and raster search patterns to cope with both large and small motions. For example, multiple diamond searches with different stride lengths ranging from 1 through 64 in multiples of 2 (i.e., Step 1, initial grid search) are firstly conducted to handle the small motion; if the motion is large (i.e., the MV with minimum SAD obtained from the previous step is larger than a pre-defined 'iRaster' value), the raster search (i.e., Step 2, zonal search) can be used to find the minimum block distortion (MBD) in the whole search range. Finally, a fine refinement step (i.e., Step 3) is used to iteratively refine the MVs obtained from the previous step, by using either raster refinement or star refinement (square/diamond patterns).

In contrast to the full search, TZ Search can alleviate the

computational burden without degrading video quality remarkably. However, it still does not make full use of the motion characteristics of surveillance videos. For example, TZ Search will search at least 22 points even if the actual MVD is zero (e.g., for static regions in the scene).

2.3. Early termination of motion search

Instead of utilizing fast search patterns and adaptive search range, the other way is to terminate the search process and the corresponding ME calculation early. By considering that a signicant portion of blocks have a zero-MV after ME, the zero-motion-termination method was proposed in Yang et al. (2002) by comparing the SAD with a threshold at the Zero-MV point to determine whether to early terminate the search procedure. Following this idea, an early-termination method was proposed in Yang et al. (2005) by estimating two thresholds for different block sizes. Similarly, an early termination algorithm was proposed in Sarwer and Wu (2009), with an adaptive threshold based on the statistical characteristics of the RD cost regarding the current block and previously-processed blocks.

To accelerate the TZ Search in HEVC, an Early-Termination TZ Search algorithm (ETTZ) was proposed in Pan et al. (2013). Following the observation that the minimum distortion point (MDP) is often central-biased and the median predictor (MP) is more likely to be the best MV, ETTZ tries to early terminate the TZ Search process: A diamond search with radius of 1 (for small CUs) or a hexagon search with radius of 2 (for large CUs) is firstly conducted with the MP as the center so as to test whether it is the MDP; If so, the TZ Search procedure is skipped and this MP is set as the MVP, otherwise the search is conducted with the MDP as the start point. Their experiments showed that it could significantly save the encoding time, with the ignorable degradation in the RD performance. However, it is easy to fall in the local optimum, because although the MP is the MDP among the neighboring diamond and hexagon points, it is probably not the MDP among the points that the original TZ Search can reach.

2.4. Optimization of CU Mode Decision

HEVC introduces a quad-tree division structure to represent variable-size CUs, which makes the motion search more precise and more flexible. Meanwhile, it is more time-consuming because before coding a $2N \times 2N$ region, it needs to determine whether this region should be encoded as a whole $2N \times 2N$ CU or recursively encoded in the form of four separate parts. This process requires recursively calculating the RD cost for each kind of partitions. In order to reduce the number of candidate CU blocks and correspondingly save the encoding time, various optimization algorithms of CU mode decision were proposed in recent years. The main idea is to skip mode decision and decide the CU size early. Hu et al. (2015) proposed a fast mode decision algorithm based on the Neyman-Pearson rule, which consists of early-skipped mode decision and fast CU size decision. Xiong et al. (2015) proposed a fast inter CU decision based on the latent SAD estimation. In Xiong et al. (2014a), a fast CU decision based on Markov random field (MRF) was proposed for HEVC inter frames. A pyramid motion divergence (PMD) based method was proposed in Xiong et al. (2014b) to early skip the specific inter CUs in HEVC. Different from the above approaches, Zupancic et al. (2016) used the adaptive coding unit visiting order to optimize inter-prediction for video coding.

These optimization algorithms can effectively reduce the encoding time. However, they could be further optimized for surveillance videos. For example, Zupancic et al. (2016) will perform four CU depth levels with CU sizes ranging from 8×8 to 64×64 even if the final CU mode is determined as a whole of 64×64 (e.g., for static regions in the scene).

2.5. Motion search for surveillance video coding

It is obvious that surveillance video has some special characteristics that can be exploited by fast motion search methods. For example, the search procedure can be significantly simplified in static background regions, without suffering the risk of degrading the RD performance. Towards this end, Wang and Dong (2014) utilize the luma component of different images to segment out moving objects from background, and then selects a proper CU size for different areas. It can simplify the motion search of static background regions but will lose a lot coding efficiency due to inaccurate foreground-background separation. In Ma et al. (2015), a searching speed-up procedure was proposed for surveillance video coding by removing the rarely used prediction modes and reference frames for different CUs and PUs adaptively using both the characteristics of the surveillance video and the information of the corresponding CUs or PUs in spatial-temporal direction.

If a background picture is modeled from a set of training frames, we can utilize it to divide the current frame into background and foreground regions more precisely, and then design the fast motion search strategy for each kind of regions. Following this idea, our previous work (Zhao et al., 2014) proposed a background-foreground-division-based search (BFDS) method. In BFDS, the MVs of the background regions are set to zero to reduce a lot of search points; while for foreground regions, a modified TZ Search is adopted to achieve better coding efficiency (as shown in Fig. 2(b)): Firstly, the iterative multiple diamond searches (i.e., steps 1 and 2) are directly conducted to deal with both large and small motions while the raster search is forbidden; Then a compulsive 11×11 rectangular search (i.e., Step 3) is performed to refine the result obtained in the previous steps so as to make the ME in foreground regions more precise.

BFDS can significantly reduce the search complexity on surveillance videos. However, when the foreground regions are relatively large, the compulsory rectangular search will cause a sharp increase of search points. Moreover, there is no error-resistance mechanism for the background-foreground division. That is, if a foreground region were wrongly classified as the background one, the coding efficiency would become much worse since the zero-MV was not suitable for these motion regions. Besides, the total encoding time was not significantly reduced. To address these problems, this paper proposes a PU-Adaptive Search method (PA-search) to direct the motion search for surveillance videos in a more robust way.

3. Problem analysis

In this study, a basic assumption is that surveillance video has some special characteristics that can be exploited by fast motion search methods and CU partition. To verify this assumption, we conducted an experimental analysis on the distribution of MVs, MVDs, search strategies (i.e., using integer-pixel search only, or using both integer- and sub-pixel search) and the CU partitions for different kinds of regions in several typical surveillance sequences. This experiment was conducted on HEVC HM-16.0 and the test conditions are tabulated in Table 4. Eight surveillance videos from the PKU-SVD-A dataset (Gao et al., 2014) were used in the experiment. Among them, six are with the resolution of 720×576 (SD), and two with the resolution of 1600×1200 (HD). As shown in Fig. 3, they can be divided into different categories according to the size and moving speed of the objects. The up-left video “Office-SD” shows the working scenario inside an office with large objects and slow moving speed. The up-right three ones are about traffic surveillance, in which the vehicles are large and fast-moving. The major foreground objects in the down-left two videos are pedestrains. The down-right two ones are captured from long distance. For simplicity, here we performed the analysis on CUs rather than the finer-grained PUs (the elementary unit for prediction in HEVC) that may have the non-squared structure by asymmetric spitting.

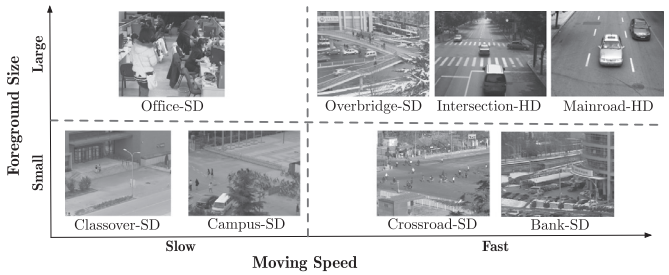


Fig. 3. Eight surveillance videos used in the analysis experiment.

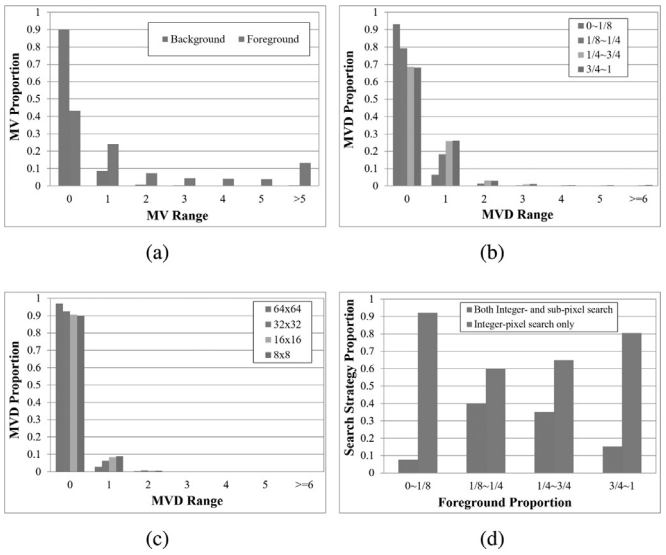


Fig. 4. (a) The MV distribution of background and foreground regions; (b) and (c) the MVD distribution based on foreground proportion and CU size; (d) the search strategy distribution based on foreground proportion.

3.1. Whether to use different search ranges for background and foreground regions or not?

Fig. 4(a) shows the statistics on the MV distribution of background and foreground regions. We can see that for background regions, more than 90% of MVs are equal to zero. This fact reveals that the background regions are totally static, and even it is unnecessary to perform motion search in these regions. We also notice that there are about 10% background regions whose MVs are equal to or larger than one. This is mainly due to the inaccurate background modeling and dynamic background (e.g., shadow, varying illumination, waving branches and leaves). Thus instead of simply setting zero-MVs to all background regions, we should introduce an error-tolerant search strategy for them. On the other hand, the MVs of foreground regions are central-biased distributed and over 20% of them are equal to or larger than 5 pixels. This means that the motion in foreground regions is relatively large and thus the corresponding search range should be larger.

3.2. What is the influence of both the foreground/background proportion in a CU and the CU size on the search strategy?

For a given CU, the final output of TZ Search is the difference between the predicted MV and the MVP (i.e., the start point), called motion vector difference (MVD). Thus we can analyze the MVD distribution for different kinds of CUs so as to examine the influence of both the foreground/background proportion in a CU and the CU size on the search strategy. The results are shown in Fig. 4 (b) and (c), where the foreground/background proportion in a CU is set into four bins, i.e., 0 ~ 1/8, 1/8 ~ 1/4, 1/4 ~ 3/4, 3/4 ~ 1, and the CU size is set from 64 × 64 to 8 × 8. Similar to the PU categories presented in the next section, CUs in the four bins can roughly correspond to Full Background CUs (FBCUs), Background CUs (BCUs), hybrid foreground-background CUs (XCUs) and Foreground CUs (FCUs), respectively.

From the two figures, we can see that the smaller the foreground proportion is and the larger the CU size is, the more MVDs distribute around zero. In Fig. 4(b), the MVDs of FBCUs tend to be zero since there are no foreground in these CUs; less MVDs equal to zero for BCUs whose foreground proportion is slightly larger than FBCUs; while for XCUs which mostly occur in the boundaries of foreground regions and for FCUs in which the majority of pixels are foreground, their MVDs are distributed more diversely and thus the search range should be larger. Similarly, Fig. 4(c) shows that when the size of a CU is large, it is more likely to output the zero MVD. This is because if there exists motion in a large CU, it will be divided into several smaller ones by the encoder so as to reduce the total RD cost of the whole region.

Table 1
The time propotion of different encoding operations with different CU sizes.

CU Size	Integer-pixel search	Sub-pixel search[a]	RD mode decision
64 × 64	19.85%	48.33%	31.82%
32 × 32	20.45%	49.69%	29.86%
16 × 16	21.88%	47.53%	30.59%
8 × 8	23.02%	38.69%	38.29%

[a] In the whole process, pixel interpolation takes about 61% time while really sub-pixel search takes about 39% time.

3.3. Whether sub-pixel search should be performed for all kinds of CUs or not?

From Table 1, we can see that sub-pixel search is more time-consuming than integer-pixel search. Basically, sub-pixel search can get a higher ME accuracy and consequently leads to better coding performance. However, it also brings higher computational complexity (totally 17 sub-pixels would be searched in HM, including 8 1/2-precision pixels and 8 1/4-precision pixels). Thus in order to reduce the total encoding time, it is necessary to optimize the sub-pixel search.

Fig. 4(d) shows the distribution of two kinds of search strategies for CUs with different foreground/background proportions, which is obtained by checking whether the final MV is equal to the result of integer- or sub-pixel motion search. We can see that for FBCUs, more than 90% of the final MVs are Integer-MVs. That is, 90% sub-pixel search in FBCUs is unnecessary. While for other CUs, we should perform both integer- or sub-pixel motion search. In this case, sub-pixel motion search should be optimized mainly by accelerating the pixel interpolation (Lin et al., 2011) since it takes about 61% time in the whole process, which is beyond the scope of this study.

3.4. Which kind of CUs should be further partitioned?

In HEVC, each CU can be further split into one, two or four PUs to specify the prediction information. So the depth of CU partition will exponentially affect the search complexity (Bross et al., 2012). Thus it is necessary to analyze the distribution of CUs that should be further partitioned. Table 2 shows the results. We can see that only 2.91/2.24/3.56% of the potential FBCUs with $N=$ 32/16/8 will be further partitioned. However, early terminating the partition of these FBCUs will cause large distortion.

We thus further analyze the relationship between two temporally consecutive FBCUs in terms of CU partition. To do so, we define S-FBCU as a special FBCU whose temporally-previous CU is also a FBCU with the same size and has not been further partitioned. Also from Table 2, we can see that only 0.20/0.28/0.43% of S-FBCUs will be further

Table 2
The propotion of the further-partitioned FBCUs, BCUs, XCUs and FCUs

CU Size	FBCU		BCU	XCU	FCU
	All	S-FBCU[a]			
64 × 64	2.91%	0.20%	14.11%	63.63%	88.89%
32 × 32	2.24%	0.28%	10.02%	50.43%	34.27%
16 × 16	3.56%	0.43%	6.21%	24.02%	29.82%

[a] Here S-FBCU denotes a special FBCU whose temporally-previous CU is also a FBCU with the same size and has not been further partitioned.

partitioned. That is, we can early terminate the CU partition for S-FBCUs, without the risk of remarkably degrading the coding efficiency.

3.5. Summary

In summary, the foreground/background proportion in a block, the block size and the depth of CU partition are three key factors that affect the search complexity in surveillance video coding. These analysis results motivate us to design adaptive search strategies for different block categories. Specifically, the zero-MV, non-sub-pixel search and CU partition early-termination should be applied to the background blocks, while different ASR prediction and early-termination search strategies should be adopted to the other blocks so as to reduce the search complexity while guaranteeing that the best MVD could fall into the search range.

4. The proposed method

4.1. Framework

The above analysis results reveal that, in order to optimize motion search for surveillance videos, the block category (or its foreground/background proportion), its size, and the depth of CU partition should be taken into account. Following this, the search optimization problem for surveillance videos can be expressed as (3).

$$(W_b^*, \varphi_b^*, d_b^*) = \underset{\substack{W\le f_W(C_b,S_b),\\ \varphi\in f_\Phi(C_b),\\ d\le f_D(C_b,S_b)}}{\operatorname{argmin}} \mathscr{L}_b(W,\varphi,d)$$
$$s.t.\ \ f_W(C_b,S_b)\le W_{Max},\ \ f_\Phi(C_b)\subseteq\Phi,$$
$$f_D(C_b,S_b)\le D(S_b),\ \text{and}\ J_\lambda^{(W,\varphi)}(b)\le T_{RD} \tag{3}$$

where $b = I(x, y, S_b)$ denotes the current block, C_b and S_b are its block category and size, $f_W(C_b, S_b)$ and $f_D(C_b, S_b)$ denote the potential search range (for both integer- and sub-pixel search) and the CU partition depth that are determined by C_b and S_b, $D(S_b)$ is the maximal CU partition depth that is related to the CU size (e.g., $D(64)=3$ for a 64 × 64 CU), $f_\Phi(C_b)$ is the set of category-related search patterns, while the meanings of $J_\lambda^{(W,\varphi)}$, and $\mathscr{L}_b$, W_{Max} and φ are the same with those in (1) and (2). Differently from the search optimization problem in (2), here three functions $f_W(C_b, S_b)$, $f_\Phi(C_b)$ and $f_\Psi(C_b)$ are introduced to determine the potential search range, the available search patterns and CU partition patterns for each block b. This can remarkably reduce the number of search points since different search ranges and patterns can be designed for different categories of blocks.

One implementation of this formulation is to automatically classify PUs and CUs into several categories and then design different search strategies (including search ranges and patterns) for PU categories and different partition strategies for CU categories. PA-Search is exactly following this idea. As shown in Fig. 5, we can see that the method works as follows:

1) A background picture is firstly trained for a SGOP and then a *background-foreground representation* (BFR) is derived for each current frame in this group using the background picture.
2) Utilizing the BFR, CUs in the current frame can be divided into four categories, {FBCUs, BCUs, XCUs, FCUs}. Similarly, PUs can also be classified into {FBPUs, BPUs, FPUs, XPUs}.
3) For FBCUs, check whether they are S-FBCUs, and if so, the partition early-termination algorithm is applied.
4) For FBPUs, zero-MV is directly assigned. To reduce the influence of PU mis-classification caused by inaccurate background modeling, an error-tolerant search strategy is also applied if the SAD distortion of a FBPU is larger than an adaptive threshold when using the zero-MV. After that, skip the sub-pixel search for those FBPUs that are still assigned with the zero-Integer-MVs.
5) For non-FBPUs, the ASR is calculated according to the PU category and size, and then the default search pattern (e.g., TZ Search in HM) can be used to obtain the final MVs. A BFR-based early termination algorithm is used to determine whether to early terminate the search.

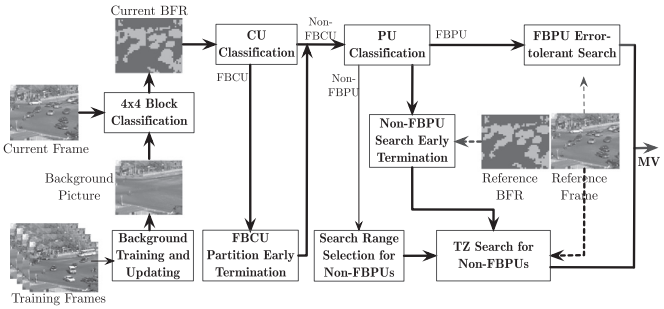


Fig. 5. The framework of PA-Search.

By adopting different search strategies for FBPUs and non-FBPUs, and taking the partition early-termination strategy for S-FBCUs, PA-Search can significantly reduce the search complexity while remaining the coding efficiency. Note that PA-Search is independent on the coding platforms, and thus can be easily implemented on either HEVC (Bross et al., 2012) or AVS2 (Dong et al., 2015).

4.2. CU and PU classification

In PA-Search, the first issue is how to classify each CU and PU into different categories according to its background-foreground distribution. This issue can boil down to three sub-problems: 1) how to perform background modeling in an online way; 2) how to derive the BFR for the current frame; and 3) how to classify its CUs and PUs using the BFR.

4.2.1. Online background modeling and updating

Generally speaking, the background modeling method in surveillance video coding should enable high coding efficiency, yet with low computational complexity. Towards this end, the fixed Gaussian Mixture Model (fGMM) (Chen et al., 2016) was utilized in PA-Search, which can achieve good coding performance and be easily implemented in hardware video codecs.

Similar to Zhang et al. (2014a), the background picture is generated *SGOP by SGOP*. That is, the first GOP in each SGOP_i is utilized as TrainSet_i to train the background picture for this SGOP, while the pictures in TrainSet_i can be coded by using the background picture in SGOP_{i-1}. In this way, each SGOP can utilize the background picture to encode its frames without delay. Note that the bit cost for coding the background pictures has been counted into the final bitrate results in our experiments.

4.2.2. BFR generation

Given the background picture, each current frame can be categorized into background and foreground regions. Then a *background-*

foreground representation (BFR) can be derived using 4 × 4 blocks (i.e., the smallest size of a block in HEVC) as the basic units (shortly as BUs).

Intuitively, for each 4 × 4 BU, if more than half pixels are foreground ones, it is a foreground block (denoted by $\mathscr{F}$), otherwise a background one (denoted by $\mathscr{B}$). This can be done by thresholding the SAD between the current BU u and its corresponding BU u' in the background picture, which can be expressed as (4).

$$C_u = \begin{cases} \mathscr{B}, & \text{if } \sum_{i=1}^{4}\sum_{j=1}^{4} |u_{i,j} - u'_{i,j}| < Th; \\ \mathscr{F}, & \text{otherwise.} \end{cases} \tag{4}$$

where $u_{i,j}$ and $u'_{i,j}$ represent the pixel values at (i, j) in u and u' respectively, and Th denotes the pre-defined threshold. In our experiments, Th is empirically set to 80 (Zhang et al., 2014b).

4.2.3. BFR-based CU/PU Classification

With the BFR, it is easy to classify PUs or CUs. Here take PU classification as the example. Let R_b denote the proportion of foreground BUs in the current PU b, then PU classification can be done according to how many 4 × 4 BUs in it belong to foreground units, which can be expressed as (5).

$$C_b = \begin{cases} \text{FBPU}, & \text{if } 0 \le R_b < \delta_1; \\ \text{BPU}, & \text{if } \delta_1 \le R_b < \delta_2; \\ \text{XPU}, & \text{if } \delta_2 \le R_b < \delta_3; \\ \text{FPU}, & \text{otherwise.} \end{cases} \tag{5}$$

where δ_1, δ_2, δ_3 are practically set to 1/8, 1/4 and 3/4. Different from two categories used in Zhao et al. (2014), here four PU categories are used, consequently enabling more delicate search strategies. Similar classification strategy can be applied to CUs.

4.3. S-FBCU partition early-termination

CU partition will severely affect the search complexity. Nevertheless, according to the analysis results in Section 3.4, we can only early terminate the CU partition for S-FBCUs. Here a S-FBCU meet two conditions: 1) both the current CU and its temporally-previous CU are FBCUs with the same size; 2) this temporally-previous FBCU has not been further partitioned. Experiments in Section 5.2 will validate the effectiveness of this S-FBCU partition early-termination strategy.

4.4. Error-tolerant search for FBPUs

For a FBPU whose foreground proportion R_b is less than 1/8, motion search can be directly skipped (namely, zero-MV is assigned). If FBPUs account for a large proportion in a surveillance video, a significant reduction in search complexity can be achieved. The reason is as follows: For TZ Search, even it is enhanced with some early termination strategy, there are at least 2 points for each PU whose RD costs need to be calculated so as to determine the starting MVP, and 20 points that need to be searched so as to find the best-matched block. While for the Zero-MV search pattern for FBPUs, the only calculation is paid for the Zero-MV distortion between the current FBPU and its zero-MV point. That is, for each FBPU, the Zero-MV search pattern can reduce 21 search points.

However, due to the inaccurate background modeling, some BPUs (or even XPUs, FPUs) may be wrongly classified into FBPUs. In this case, the Zero-MV search pattern cannot obtain the desirable RD performance. To reduce the influence of such mis-classifications, we propose an error-tolerant search algorithm for FBPUs. Its basic idea is to utilize the Zero-MV distortion to adaptively determine whether the Zero-MV search pattern can be applied to the current FBPU. To do so, a threshold T_{zero} needs to be learned and updated online. Let D_b denote the Zero-MV distortion for the current FBPU b, then this algorithm can be expressed as (6).

$$(W_b^*, \varphi_b^*) = \begin{cases} (0, \text{NONE}), & \text{if } D_b/Z_b \le T_{zero}; \\ (W^*, \varphi^*)_{\text{BPU}}, & \text{otherwise.} \end{cases} \tag{6}$$

where Z_b is the block area of b (e.g., $Z_b = S_b^2$ for a squared PU with the size of $S_b \times S_b$), (0, NONE) denotes the Zero-MV search pattern (i.e., $W_b^* = 0$ and $\varphi_b^* =$ "NONE"), and $(W^*, \varphi^*)_{\text{BPU}}$ represents the search strategy for BPUs. That is to say, if D_b/Z_b is no more than T_{zero}, the zero-MV is assigned to b; otherwise, the search strategy for BPUs, which will be described in the next subsection, can be used for b. Note that here $(W_b^*, \varphi_b^*) = (0, \text{NONE})$ means that sub-pixel search will be directly skipped. This is supported by the analysis results shown in Section 3.3 (i.e., over 90% of sub-pixel search in FBCUs is unnecessary).

Here the key issue is how to initialize and online update the threshold T_{zero}. In this study, T_{zero} is initialized using the statistical measures (i.e., mean and variance) of FBPUs' Zero-MV distortions in the TrainSet_1, where TrainSet_1 is the first training set in background modeling. Let D_i denote the Zero-MV distortion of the i^{th} FBPU in TrainSet_1, then μ and σ^2 can be expressed as (7) and (8), respectively.

$$\mu = \frac{1}{N}\sum_{i=1}^{N}\frac{D_i}{Z_i} \tag{7}$$

$$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{D_i}{Z_i} - \mu\right)^2 \tag{8}$$

where N is the number of FBPUs. Here $\frac{D_i}{Z_i}$ denotes the average distortion of each pixel in the ith FBPU. By assuming that the mis-classification results are mostly the outliners in the statistical distribution of FBPUs' Zero-MV distortions, thus we can initialize T_{zero} by following the "three sigma rule" which can be expressed as (9).

$$T_{zero} = \mu + 3\sigma. \tag{9}$$

Moreover, T_{zero} can be online updated by incrementally updating μ and σ. For the current FBPU, if its average distortion is larger than the current T_{zero}, it should be a non-FBPU; otherwise, its average distortion should be used to update μ and σ. Let $\frac{D_b}{Z_b}$ denote the average distortion for the current FBPU b, then μ and σ can be updated as (10) and (11).

$$\mu_{new} = \frac{N\mu + \frac{D_b}{Z_b}}{N+1} \approx (1-\rho)\mu + \rho\frac{D_b}{Z_b} \tag{10}$$

$$\begin{aligned} \sigma_{new}^2 &\approx \frac{(N-1)\sigma^2 + \left(\frac{D_b}{Z_b} - \mu_{new}\right)^2}{N} \\ &\approx (1-\rho)\sigma^2 + \rho\left(\frac{D_b}{Z_b} - \mu_{new}\right)^2 \end{aligned} \tag{11}$$

where ρ is a parameter to control the decay rate (ρ = 0.02 in our experiments). Thus given the initial T_{zero}, the error-tolerant search algorithm for FBPUs can be summarized in Algorithm 1.

4.5. Adaptive search for non-FBPUs

Typically, different categories of non-FBPUs also exhibit different motion characteristics. Thus to reduce the search complexity, an adaptive search range (ASR) selection strategy is proposed for non-FBPUs. Moreover, a BFR-based early-termination algorithm is also proposed to determine whether to early terminate the search process for non-FBPUs or not.

4.5.1. ASR selection

According to the analysis results shown in Fig. 4 (b) and (c), we can design an ASR selection strategy for non-FBPUs, as (12).

Input: The current FBPU $b = I(x, y, S_b)$; the initial threshold T_{zero}.
Output: b's MV (u, v); RD cost $J^*_\lambda(b)$; and the updated T_{zero}.

procedure

1. Calculate the Zero-MV distortion.

$$D_b = \mathcal{D}(I(x, y, S_b) - F(x, y, S_b))$$

where $\mathcal{D}(\cdot)$ denotes the block matching measure, and F denotes the reference frame.

2. Determine the search range and pattern.

Calculate Z_b by S_b, where $Z_b = S_b^2$ for a squared PU;

if $D_b/Z_b \le T_{zero}$ **then**

$(u, v) = (0, 0)$; $J^*_\lambda(b) = D_b$;

Skip sub-pixel search for b;

3. Update the threshold.

Update μ and σ using (10) and (11) ;

Update T_{zero} using (9) ;

else

Treat b as a BPU and perform the corresponding motion search strategy $(W^*, \varphi^*)_{\mathrm{BPU}}$, return MV (u, v) and the optimal RD cost $J^*_\lambda(b)$.

end if

end procedure

Algorithm 1. The error-tolerant search algorithm for FBPUs

$$W_b^* = \begin{cases} 1, & \text{if } C_b = \text{BPU and } Z_b \geq \beta; \\ W_{\text{STD}}/4, & \text{if } C_b = \text{BPU and } Z_b < \beta; \\ W_{\text{STD}}/2, & \text{if } C_b = \text{FPU and } Z_b \geq \beta; \\ W_{\text{STD}}, & \text{if } (C_b = \text{FPU and } Z_b < \beta) \text{ or } (C_b = \text{XPU and } Z_b \geq \beta); \\ 1.5W_{\text{STD}}, & \text{if } C_b = \text{XPU and } Z_b < \beta. \end{cases} \quad (12)$$

where W_{STD} denotes the standard search range when applying the default fast search pattern to this PU, and β is a threshold ($\beta = 1024$ in our experiments). Note that, if $W_{\text{STD}} = 64$ (i.e., the standard search window is 64 × 64), $W_{\text{STD}}/4$ should be 16.

Basically, two principles are considered when designing this ASR selection strategy: First, for PUs with the same size, W_b^* will gradually increase from BPUs, to FPUs, and then to XPUs. For BPUs whose foreground proportion is slightly larger than FBPUs, less MVDs are equal to zero and thus small non-zero search ranges should be used. For FPUs, we can directly apply the default fast search pattern and thus follow the standard search range W_{STD}. While XPUs, which mostly occur in objects' boundaries, should be assigned to a larger search range than W_{STD}. This is mainly due to the fact that different motion properties of background and foreground pixels in an XPU will cause a large prediction distortion.

Second, the PU size should also be taken into account. Within the same PU category, the larger the PU size is, the smaller its search range should be used. This is because if there exists motion in a large PU, it would be divided into smaller ones by the encoder so as to reduce the total RD cost of the whole region. Here a large PU means that its area Z_b should be no smaller than 1024 (i.e., a 32 × 32 PU). Following this, $W_b^* = 1$ for a large BPU while $W_{\text{STD}}/4$ for a small BPU; similarly, W_b^* is set to $W_{\text{STD}}/2$ for a large FPU, W_{STD} for a small FPU or a large XPU, while $1.5W_{\text{STD}}$ for a small XPU.

This ASR selection strategy can also be summarized in Table 3. Note that, if a search range is assigned to a non-FBPU (e.g., $W_{\text{STD}}/2$ for a 32 × 32 FPU), it indicates that the default fast search pattern (e.g., TZ Search) should be conducted within the corresponding search window. By using this strategy, the encoder is able to have more chances to find the best MVs while avoiding some unnecessary search points.

4.5.2. Search early-termination for non-FBPUs

To further reduce the search complexity for non-FBPUs, an early-termination strategy can be implemented by comparing the BFRs of the current frame and the reference frame. Fig. 6 shows an example. Here the reference frame may be either a recent/hierarchical reference frame or a background picture. For a non-FBPU, the search is initially conducted within a rectangular search window in the reference frame. However, for some candidate blocks in the window, we actually need not to perform all search steps (namely, the search can be skipped). The reason is as follows: Usually, a non-FBPU is used to represent a part of an object in the current frame. It is certain that this object should also occur as foreground in one of the reference frames. In this case, the best-matched block should have the same or at least highly similar FBR as the current non-FBPU. In other words, if the FBR of a candidate block is dissimilar with that of the current non-FBPU, the search can be skipped directly.

The remaining problem is how to compare the FBRs between the current non-FBPU and each candidate block in the reference frames through a low-complexity way. To quickly filter out the candidate blocks whose BFRs are not similar to that of the current non-FBPU, PA-Search utilizes a simple point-based block filtering strategy. As Fig. 7 shows, four corner points in both the candidate block and the current non-FBPU are selected. Let $b = I(x, y, S_b)$ denote the current non-FBPU, S_{bx} and S_{by} are its width and depth, then the four points in b can be expressed as $I(x + i, y + j)$ where $(i, j) \in \{(0, 0), (0, S_{bx}), (S_{by}, 0), (S_{by}, S_{bx})\}$. Then the BFR values of the four point pairs are compared to evaluate whether they are matched. If the number of the matched points is larger than a pre-defined threshold T_{match}, this candidate block should be searched; otherwise, it could be skipped. In Fig. 7, for example, the numbers of the matched points in (b) and (c) are 3 and 2 respectively. If T_{match} is set as 2, the candidate block in (b) will be searched while that in (c) will be skipped. In this study, T_{match} will be set experimentally.

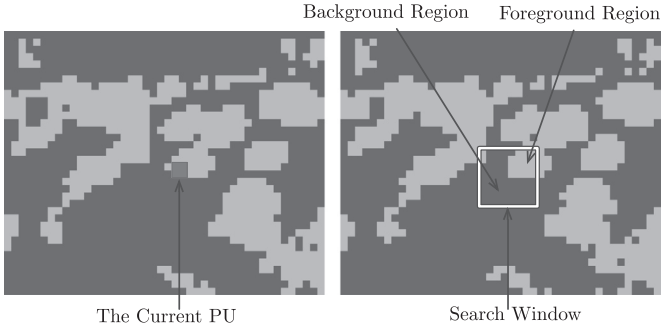
By summarizing the ASR selection strategy and the BFR-based early-termination algorithm, the motion search procedure for non-FBPUs can be discribed in Algorithm 2.

Table 3
The search range selection strategy for non-FBPUs

PU category	PU size	
	64 × 64 ~ 32 × 32 ($Z_b \geq 1024$)	32 × 32 ~ 4 × 4 ($Z_b < 1024$)
BPU	1	$W_{\text{STD}}/4$
FPU	$W_{\text{STD}}/2$	W_{STD}
XPU	W_{STD}	$1.5W_{\text{STD}}$

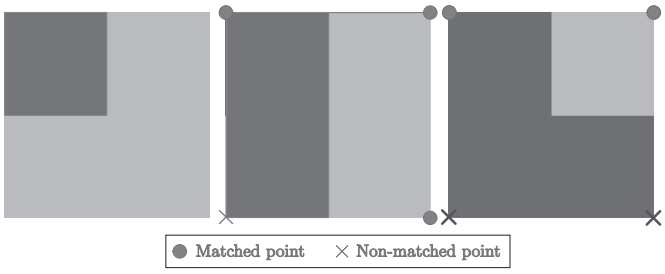
Table 4
Test conditions.

Max. CU size	64 × 64
Max. CU depth	4
Asymmetric PU (AMP) partitions	enabled
Quantization Parameter (QP)	22, 27, 32, 37
Search Range	[-64, 64]
Configuration file	encoderlowdelayPmain.cfg
Number of tested frames	2000



(a) BFR of the Current Frame (b) BFR of the Reference Frame

Fig. 6. The BFRs of the current frame and its reference frame. For the current PU, only the foreground regions within the search window will be searched.



(a) Prediction Unit (b) Matched Block (c) Non-matched Block

Fig. 7. An example of the point-based block filtering strategy. The regions in light color are foreground while those in deep color are background. If the corresponding points have the same BFR value, they are called as two matched points; then if the number of the matched points between two blocks is larger than a threshold T_{match} (here $T_{match} = 2$), they are called as matched blocks.

4.6. Complexity analysis

This subsection will briefly describe the complexity analysis of PA-

Input: The current FBPU $b = I(x, y, S_b)$ with its category C_b; the BFR of the current frame I, $\Upsilon(I)$; the reference frames $\{F_k\}_1^K$ with their corresponding BFRs $\{\Upsilon(F_k)\}$; the threshold T_{match}.
Output: b's MV (u, v), RD cost $J_\lambda^*(b)$.

procedure
1. Calculate the search range W_b^* for b using (12);
$J_\lambda^*(b)$=0;
for $\forall F_k, k = 1, \cdots, K$ **do**
2. Calculate the search order set $\varphi(x, y)$ within W_b^*, using TZ Search;
for $\forall c \in \varphi(x, y)$ where $c = F_k(s, t, S_b)$ **do**
3. Match between BFRs of b and c;
$M(b, c)$=0;
for $(i, j) \in \{(0, 0), (0, S_{bx}), (S_{by}, 0), (S_{by}, S_{bx})\}$ **do**
if $\Upsilon(I, x + i, y + j) = \Upsilon(F_k, s + i, t + j)$ **then**
$M(b, c) = M(b, c) + 1$;
end if
end for
4. Search with BFR-based early-termination
if $M(b, c) \leq T_{match}$ **then**
Skip the search on c; continue;
else
Calculate $J_\lambda^{(W,\varphi)}(b)$ using (1);
$J_\lambda^*(b) \leftarrow \min\{J_\lambda^*(b), J_\lambda^{(W,\varphi)}(b)\}$;
end if
end for
end for
5. Return the MV (u, v) corresponding to $J_\lambda^*(b)$.
end procedure

Algorithm 2. The motion search procedure for non-FBPUs

Search over the anchor (i.e., TZ Search). More details can be found in the Appendix.

Statistically, two factors will significantly affect the actual search complexity if TZ Search or PA-Search is used on a given surveillance video V, i.e., the foreground proportion R_v and the motion factor m_v. Here R_v is calculated by counting the proportion of foreground regions in the BFRs of that video, thereby reflecting the scene complexity; while m_v is defined as the proportion of PUs in the video whose MVs are greater than the pre-defined value r, thus reflecting the average motion speed of foreground objects. With these assumptions, the total number of search points in TZ Search can be approximated as

$$N_{TZ}(R_v, m_v) = 22 + 121.4R_v + 169R_v m_v. \tag{13}$$

Meanwhile, since PA-Search will adopt different search strategies for different PU categories, we will introduce three additional variables R_B, R_F and R_X to denote the percentages of three non-FBPUs (i.e., BPUs, FPUs and XPUs), with $R_B + R_F + R_X = 1$. Then the total number of search points in PA-Search can be approximated as

$$\begin{aligned} N_{PA}(R_v, m_v, R_F, R_X) &= 1 + R_v(106.88 + 35.52R_F + 53.28R_X) \\ &\quad + 10.563R_v m_v(1 + 15R_F + 35R_X). \end{aligned} \tag{14}$$

Finally, we can define the Search-Points-Proportion value (SPP-value) as the ratio of the number of search points in PA-Search divided by that in the anchor, as

$$\begin{aligned} SPP &= \frac{N_{PA}(R_v, m_v, R_F, R_X)}{N_{TZ}(R_v, m_v)} \\ &= \frac{\begin{pmatrix} 1 + R_v(106.88 + 35.52R_F + 53.28R_X) \\ +10.563R_v m_v(1 + 15R_F + 35R_X) \end{pmatrix}}{22 + 121.4R_v + 169R_v m_v} \end{aligned} \tag{15}$$

This SPP-value can be used to approximately estimate the reduction of search complexity by using the proposed fast motion search method. In the experiments, we will compare the estimated proportion with its actual value so as to validate its estimation precision.

5. Experiments

5.1. Experimental settings

In this section, several experiments were conducted to validate the effectiveness of the proposed PA-Search. The main objectives were two-fold: (1) to explore how different components of PA-Search work, and (2) to demonstrate the advantage of PA-Search over several state-of-the-art methods.

Totally sixteen uncompressed surveillance sequences from the PKU-SVD-A dataset[1] were used in the experiments. These sequences were captured from different surveillance scenes (e.g., campus, office, road intersection, etc.), with large or small objects (LO/SO), and fast or slow motion (FM/SM). Here they are divided into two subsets: Subset-1 with eight SD ~ HD sequences that have been used in Section 3 for problem analysis (as shown in Fig. 3), and Subset-2 with the other eight 1080P videos that were mostly captured with shadow/dim illumination conditions (as shown in Fig. 8). Note that sequences in Subset-1 have also been used in (Zhang et al., 2014a) for surveillance video coding experiments.

Three metrics were used to evaluate the performance of different search methods, i.e. Bjontegaard Distortion (BD)-rate (Bjontegard, 2001), Search-Points-Proportion value (SPP-value, including the points searched in sub-pixel search) and Total-Encoding-Time-Proportion value (TETP-value). By combining the bitrate and the PSNR, the BD-rate value reflects the bitrate difference under the same PSNR. The negative value of BD-rate means that there is bit-saving over the anchor; otherwise, there is some loss in coding efficiency. On the other side, the SPP-value (TETP-value) is defined as the ratio of search points (the total encoding time) in the given search method divided by that in the anchor. Therefore, BD-rate can be used to measure the coding efficiency while SPP-value and TETP-value can be used to measure the coding complexity.

[1] The PKU-SVD-A dataset, http://pkuml.org/resources/pku-svd-a.html.

Fig. 8. The representative frames for the surveillance videos in Subset-2 of the PKU-SVD-A dataset.

The experiments were conducted on the recent stable version of HEVC reference software, HM-16.0, with TZ Search as the anchor. The test conditions are tabulated in Table 4. The other coding parameters adopts the default setting in the HM-16.0 coding profile. The hardware platform is Intel Xeon CPU e5-1620 v2 @ 3.70GHz, 16.0GB RAM with the Microsoft Windows 7 64-bit OS.

5.2. How it works

5.2.1. Parameter selection

In PA-Search, some parameters need to be selected in advance, such as the matching threshold T_{match} for non-FBPU search early-termination. Thus in the first set of experiments, our objective was to determine them experimentally on the eight sequences in Subset-1. For the other parameters, some can be determined according to the experimental analysis in Section 3 (e.g., $\delta_{1\sim3}$, β), while some others can be set empirically (e.g., ρ). Therefore, we do not include the experiments for them here.

In the process of generating BFR, the threshold Th determines if a BU is a foreground block or not. Fig. 9 shows the distribution of the difference of pixel value between current frame and background frame. As we know, there are many static background regions in surveillance videos. From Fig. 9 we can see that most difference of pixel value are within 5, which means if the difference of one pixel less than 5 then it is a background pixel. Considering that the size of BU is 4×4, the value of Th is setted to 80 (5×16).

In the non-FBPU early-termination algorithm, the matching threshold T_{match} determines which candidate blocks can be skipped for search. Within the value range $[0, \cdots, 4]$, a larger value of T_{match} means a tighter matching criterion such that few candidate blocks will be searched. Thus this experiment was to determine its optimal value in terms of coding efficiency and search complexity, where the anchor was $T_{match} = 0$ (i.e., no early-termination for non-FBPUs).

Fig. 10 shows the BD-rates and SPP-values when using different values of T_{match}. We can see that when increasing the value of T_{match}, the coding efficiency becomes worse while the search complexity reduces synchronously. This is reasonable since with fewer blocks to be searched, there should be more PUs whose best-matched blocks in the reference frames cannot be found, and consequently the total RD cost would increase inevitably. We also notice that when $T_{match} = 2$, the BD-rate is less than 0.15% while the increase of SPP-value is no more than 20% compared with the no early-termination case. Moreover, its BD-rate is almost the same as that of $T_{match} = 1$, but its search complexity is less by about 5%. When T_{match} is larger than 2, the coding efficiency declines rapidly. Therefore, it seems that the optimal value of T_{match} is 2.

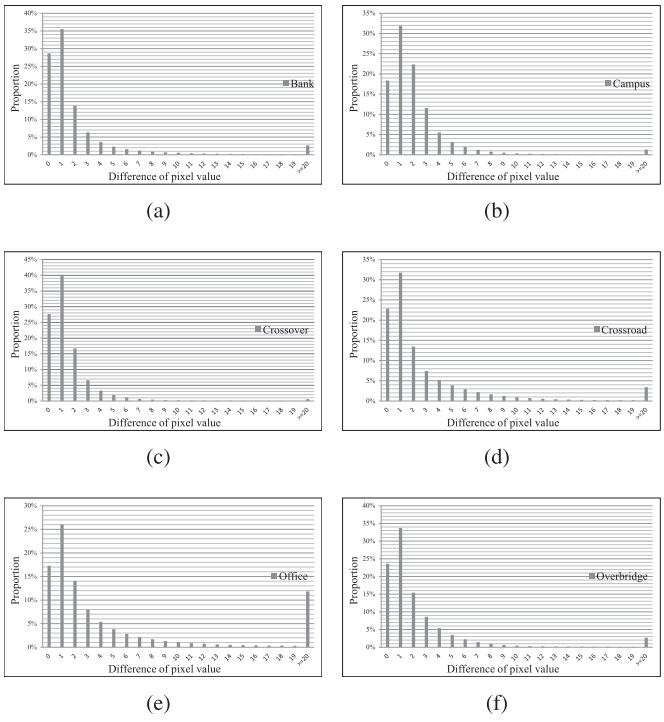


Fig. 9. The distribution of the difference of pixel value between current frame and background frame.

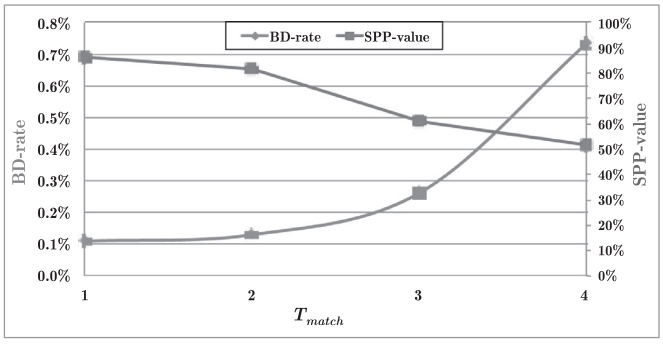


Fig. 10. The BD-rate and SPP-value curves for different values of T_{match}, where the anchor is $T_{match}=0$ (i.e., no early-termination for non-FBPUs).

5.2.2. Contributions of different components in PA-Search

PA-Search can be roughly divided into two parts: (1) CU-part, including S-FBCU partition early-termination, and (2) PU-part, including zero-MV and error-tolerant search search for FBPUs, ASR selection and BFR-based search early-termination for non-FBPUs. So this experiment was to evaluate the influence of the two parts on the coding efficiency and complexity.

Table 5 shows the results. We can see that, the BD-rate of PU-part over the anchor (i.e., HM-16.0) is slightly larger than CU-part + PU-part over PU-part. On the contrary, the SPP-value saving of PU-part is remarkably larger than that of CU-part (60.10% vs. 26.05%). This indicates that in PA-search, PU-part is the main component that contributes to the reduction of search points. We also notice that their TETP-values are comparable (28.74% vs. 28.79%). This is because CU-part can effectively reduce the depth of CU partition, where each CU partition step consists of several computationally-expensive operations such as integer-pixel search, sub-pixel search and RD-model decision. In this case, a small number in the reduction of search points by CU-part would incur a large saving of the total encoding time.

5.3. Comparison with the State-of-the-Arts

This subsection describes the experiments that were designed to compare the performance of PA-Search and state-of-the-art methods on surveillance videos. Two state-of-the-art methods were involved in the experiments, including BFDS (Zhao et al., 2014) and Ma et al. (2015). Note that both of them can also be seen as the modified versions of TZ Search.

Table 6 shows the results. We can see that, compared with the original TZ Search, the BD-rate of PA-Search has a negligible loss (averagely 0.70% on all surveillance videos) but its coding complexity can be reduced by averagely 66.90% of SPP-value saving and 46.69% of TETP-value saving, respectively. We can also find that on surveillance videos, PA-Search significantly outperforms the two state-of-the-art methods, BFDS and Ma et al. (2015), in terms of both SPP-value and TETP-value. On average, PA-Search has slightly higher BD-rate than BFDS (0.70% vs. 0.53%) and lower BD-rate than Ma et al. (2015) (0.70% vs. 1.57%), but much larger SPP-value saving (66.90% vs. 24.59% and 47.29%) and TETP-value saving (46.69% vs. 3.01% and 34.10%). The results show that PA-Search is able to reduce the coding complexity considerably while maintaining the coding efficiency.

Noticeably, Ma et al. (2015) can reduce the search complexity remarkably by removing the rarely used prediction modes and reference frames for different CUs and PUs adaptively. However, it will bring lots of loss in performance because the condition estimating which prediction modes or reference frames should be removed is too simply. It is easy for the mv to fall in the local optimum. As the previous work of PA-Search, BFDS can reduce the search complexity on surveillance videos, with 24.59% of the average SPP-value saving. However, due to the lack of non-sub-pixel search mechanism for FBPUs and CU partition early-termination strategy for S-FBCUs, the total encoding time will not reduce significantly. This is the reason why BFDS has a small TETP-value saving (averagely 3.01%). Clearly, these drawbacks have been successfully solved by PA-Search.

5.4. Supplementary experiment

In Section 4.5, we present the complexity analysis for PU-part of PA-Search. However, one may argue the precision of this analysis method since it is built on several assumptions (e.g., ignoring the influence of the BFR-based early-termination algorithm) and with some approximate values. Thus this experiment was designed to validate its applicability.

Table 7 shows the comparison results. We can see that the average difference between the estimated and the actual SPP-values of PU-part of PA-Search is 3.77% for surveillance videos. This indicates the complexity analysis method for PA-Search is reasonably accurate.

We also notice that the estimated SPP-values are always larger than the actual values more or less. This is because by ignoring the influence of the block size on the search range selection and the possible speed-up of the BFR-based early-termination algorithm, the approximate estimation method tends to use the maximum number of search points as the estimated value. In this sense, the estimated value can be seen as the worst case of search points in PA-Search.

6. Conclusions

This paper proposes a PU-Adaptive Search (PA-Search) method for surveillance videos, by adaptively adopting search strategies for different CU and PU categories. Experimental results on the PKU-SVD-A dataset demonstrate the advantage of PA-Search on the HEVC reference software HM-16.0. In particular, PA-Search can reduce the number of search points of 66.90% and total encoding time of 46.69% over TZ Search on HM-16.0, while maintaining the coding efficiency.

In the future work, we will further optimize the performance of PA-Search and then extend it to scene videos, a super-set of surveillance

Table 5
The coding efficiency and complexity of CU-part and PU-part of PA-Search on HM-16.0.

Sequence		PU-part vs. Anchor			PU-part + CU-part vs. PU-part		
		BD-rate (%)	SPP-value Saving (%)	TETP-value Saving (%)	BD-rate (%)	SPP-value Saving (%)	TETP-value Saving (%)
Subset-1 of PKU-SVD-A	BankSD	0.61	56.87	27.13	0.24	12.26	22.02
	ClassoverSD	0.89	68.72	33.48	0.04	19.85	31.10
	CompusSD	0.86	64.54	31.65	0.09	17.96	26.32
	CrossroadSD	0.73	46.54	18.24	0.04	6.96	10.72
	OfficeSD	0.25	36.87	17.30	0.04	6.90	12.05
	OverbridgeSD	0.81	54.75	25.44	0.08	14.30	18.80
	IntersectionHD	0.53	46.00	23.58	0.22	15.09	22.65
	MainroadHD	0.55	53.11	27.82	-0.05	16.87	26.92
Subset-2 of PKU-SVD-A	JingChunRoadnorth	0.22	62.39	27.01	0.51	55.09	33.76
	JingChunRoadsouth	0.44	79.05	42.19	0.35	45.13	47.59
	RedHouseNo.1east	0.78	86.27	45.67	0.03	26.73	56.31
	RedHouseNo.1south	0.36	74.62	37.82	0.34	34.21	39.71
	RedHouseNo.4north	0.18	65.90	31.05	0.26	40.86	33.34
	RedHouseNo.4west	0.31	74.60	37.33	0.31	49.40	40.36
	WeiMingLakeeast	0.37	58.63	23.53	0.36	51.06	30.72
	HaiDiancrossroad	0.41	32.75	10.58	0.03	4.18	8.34
AVG		**0.52**	**60.10**	**28.74**	**0.18**	**26.05**	**28.79**

Table 6
The coding efficiency and coding complexity of PA-Search, BFDS and Ma et al. (2015) on HM-16.0, with the original TZ Search as the anchor.

Sequence		PA-Search			BFDS			Ma et al. (2015)		
		BD-rate (%)	SPP-value Saving (%)[a]	TETP-value Saving (%)[a]	BD-rate (%)	SPP-value Saving (%)	TETP-value Saving (%)	BD-rate (%)	SPP-value Saving (%)	TETP-value Saving (%)
Subset-1 of PKU-SVD-A	BankSD	0.85	59.75	40.66	0.68	19.95	1.79	1.51	48.09	36.66
	ClassoverSD	0.92	72.88	51.80	0.93	28.75	3.30	2.84	62.67	47.61
	CompusSD	0.95	68.08	46.86	0.97	30.19	3.44	1.33	53.72	40.04
	CrossroadSD	0.77	49.18	26.39	0.73	18.51	2.14	1.44	37.88	20.37
	OfficeSD	0.29	40.39	26.72	0.32	4.22	0.79	1.09	39.81	24.93
	OverbridgeSD	0.90	58.37	37.74	0.81	26.50	3.42	1.94	52.32	32.19
	IntersectionHD	0.75	53.69	40.52	0.56	21.18	2.19	1.78	42.14	30.43
	MainroadHD	0.50	58.22	43.33	0.72	42.05	4.22	1.56	39.29	27.08
Subset-2 of PKU-SVD-A	JingChunRoadnorth	0.73	79.16	50.61	0.25	23.94	2.93	1.47	45.53	37.32
	JingChunRoadsouth	0.80	86.25	66.95	0.30	28.98	4.45	1.52	42.72	33.07
	RedHouseNo.1east	0.82	92.26	74.25	0.79	29.63	4.22	1.49	54.69	40.11
	RedHouseNo.1south	0.71	85.29	62.20	0.29	26.77	3.60	1.61	47.38	38.42
	RedHouseNo.4north	0.44	74.62	53.56	0.20	24.57	3.11	1.22	49.89	37.41
	RedHouseNo.4west	0.62	82.43	61.25	0.35	26.68	3.14	1.34	60.09	44.49
	WeiMingLakeeast	0.74	74.52	46.34	0.19	23.50	3.24	1.79	52.47	33.84
	HaiDiancrossroad	0.44	35.23	17.81	0.32	18.07	2.23	1.20	27.89	21.63
AVG		**0.70**	**66.90**	**46.69**	**0.53**	**24.59**	**3.01**	**1.57**	**47.29**	**34.10**

[a] Here the value is equal to 1 minus the corresponding SPP-value (or TETP-value). The same below.

Table 7
Comparison of the estimated and the actual SPP-values of the PU-part of PA-Search by using TZ Search as the anchor.

Sequence		Estimated (%)	SPP-value (%)	Diff. (%)
Subset-2 of PKU-SVD-A	JingChunRoadnorth	41.94	37.61	4.33
	JingChunRoadsouth	26.25	20.95	5.30
	RedHouseNo.1east	18.17	13.73	4.44
	RedHouseNo.1south	28.56	25.38	3.18
	RedHouseNo.4north	37.28	34.10	3.18
	RedHouseNo.4west	27.50	25.40	2.10
	WeiMingLakeeast	47.58	41.37	6.21
	HaiDiancrossroad	68.63	67.25	1.38
	AVG	**36.99**	**33.22**	**3.77**

videos that are often captured on a relatively-fixed place by (mostly fixed-view) cameras for a long time (e.g., classroom videos and meeting videos). Moreover, we also plan to integrate the proposed PA-Search method in the open-source fast HEVC encoder, X.265.

Acknowledgments

This work is partially supported by grants from the National Basic Research Program of China under grant 2015CB351806, the National Natural Science Foundation of China under contract nos. U1611461, 61390515, and 61425025.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.cviu.2018.02.009.

References

Bjontegard, G., 2001. Calculation of average PSNR differences between rd-curves. ITU-T VCEG-M33.

Bossen, F., Bross, B., Suhring, K., Flynn, D., 2012. HEVC complexity and implementation analysis. IEEE Trans. Circuits Syst. Video Technol. 22 (12), 1685–1696.

Bross, B., Han, W.-J., Sullivan, G.J., Ohm, J.-R., Wiegand, T., 2012.. High efficiency video coding (HEVC) text specification draft 9. JCTVC-K1003, Joint Collaborative Team on Video Coding (JCT-VC), Stockholm, Sweden. document.

Chen, W., Tian, Y., Wang, Y., Huang, T., 2016. Fixed-point gaussian mixture model for analysis-friendly surveillance video coding. Comput. Vis. Image Understand. 142, 65–79.

Chen, Z., Song, Y., Ikenaga, T., Goto, S., 2007. Macroblock level adaptive search range algorithm for variable block size motion estimation in H.264/AVC. Proceedings of International Symposium Intelligent Signal Processing and Communication Systems. pp. 598–601.

Chen, Z., Zhou, P., He, Y., 2003. Hybrid unsymmetrical-cross multi-hexagon-grid search strategy for integer pel motion estimation in H.264. Proceedings of Picture Coding Symposium. pp. 17–22.

Chung-Cheng Lou, S.-W.L., Kuo, C.-C.J., 2010. Adaptive search range selection in motion estimation. Proceedings of IEEE Conference on Acoustics Speech and Signal Processing. pp. 918–921.

Dhara, B.C., Saha, S.K., Chanda, B., 2010. A video coding technique using octagonal motion search and BTC-PF method for fast reconstruction. Proceedings of AST/UCMA/ISA/ACN 2010, LNCS 6059. pp. 480–490.

Dong, S., Tian, Y., Huang, T., 2015. Performance evaluation for avs2 scene video coding techniques. Multimedia Big Data (BigMM), 2015 IEEE International Conference on. IEEE, pp. 411–414.

Gao, W., Tian, Y., Huang, T., Ma, S., Zhang, X., 2014. The IEEE 1857 standard empowering smart video surveillance systems. IEEE Intell. Syst. 29 (5), 30–39. Sep.-Oct.

Girod, B., 1994. Rate-constrained motion estimation. SPIE Proceedings of Visual Communication and Image Processing. 2308. pp. 1026–1034.

Hu, Q., Zhang, X., Shi, Z., Gao, Z., 2015. Neyman-Pearson-based early mode decision for HEVC encoding. IEEE Trans. Multimedia 18 (3), 1.

Kossentini, F., Lee, Y.-W., Smith, M.J.T., Ward, R.K., 1997. Predictive RD optimized motion estimation for very low bit-rate video coding. IEEE J. Sel. Areas Commun. 15 (9), 1752–1763.

Lin, W., Panusopone, K., Baylon, D.M., Sun, M.-T., Chen, Z., Li, H., 2011. A fast sub-pixel motion estimation algorithm for h.264/avc video coding. IEEE Trans. Circuits Syst. Video Technol. 21 (2), 237–242.

Lin, Y.-C., Tai, S.-C., 1997. Fast full-search block-matching algorithm for motion-compensated video compression. IEEE Trans. Commun. 45 (5), 527–531.

Lou, C.-C., Lee, S.-W., Kuo, C.-C.J., 2010. Adaptive motion search range prediction for video encoding. IEEE Trans. Circuits Syst. Video Techn. 20 (12), 1903–1908.

Ma, L., Qi, H., Zhu, S., Ma, S., 2015. A fast background model based surveillance video coding in HEVC. Visual Communications and Image Processing Conference. pp. 237–240.

Pan, Z., Yun Zhang, S.K., Wang, X., Xu, L., 2013. Early termination for TZ search in HEVC motion estimation. Proceedings of IEEE Conference on Acoustics Speech and Signal Processing. pp. 1389–1393.

Paul, A., Wang, J.-F., Yang, J.-F., 2008. Adaptive search range selection for scalable video coding extension of H.264/AVC. Proceedings of IEEE Region 10 Conference. pp. 1–4.

Sarwer, M.G., Wu, Q.M.J., 2009. Adaptive variable block-size early motion estimation termination algorithm for H.264/AVC video coding standard. IEEE Trans. Circuits Syst. Video Technol. 19 (8), 1196–1201.

Shen, L., Zhang, Z., Liu, Z., 2014. Adaptive inter-mode decision for hevc jointly utilizing inter-level and spatiotemporal correlations. IEEE Trans. Circuits Syst. Video Technol. 24 (10), 1709–1722.

JVT of ISO/IEC MPEGITU-T VCEG, 2010. MVC Software Reference Manual-JMVC 8.2, May.

Wang, J., Dong, L., 2014. An efficient coding scheme for surveillance videos based on high efficiency video coding. Natural Computation (ICNC), 2014 10th International Conference on. pp. 899–904.

Xing, P., Tian, Y., Zhang, X., Wang, Y., Huang, T., 2013. A coding unit classification based AVC-to-HEVC transcoding with background modeling for surveillance videos. Visual Communications and Image Processing. pp. 1–6.

Xiong, J., Li, H., Meng, F., Wu, Q., 2015. Fast hevc inter cu decision based on latent sad estimation. IEEE Trans. Multimedia 17 (12), 2147–2159.

Xiong, J., Li, H., Meng, F., Zhu, S., Wu, Q., Zeng, B., 2014. MRF-based fast HEVC inter cu decision with the variance of absolute differences. IEEE Trans. Multimedia 16 (8), 2141–2153.

Xiong, J., Li, H., Wu, Q., Meng, F., 2014. A fast hevc inter cu selection method based on pyramid motion divergence. IEEE Trans. Multimedia 16 (2), 559–564.

Yang, J.-F., Chang, S.-C., Chen, C.-Y., 2002. Computation reduction for motion search in low rate video coders. IEEE Trans. Circuits Syst Video Technol. 12 (10), 948–951.

Yang, L., Yu, K., Li, J., Li, S., 2005. An effective variable block-size early termination algorithm for H.264 video coding. IEEE Trans. Circuits Syst. Video Technol. 15 (6), 784–788.

Zhang, X., Huang, T., Tian, Y., Gao, W., 2014. Background-modeling based adaptive prediction for surveillance video coding. IEEE Trans. Image Process. 23 (2), 769–784.

Zhang, X., Huang, T., Tian, Y., Geng, M., Ma, S., Gao, W., 2013. Fast and efficient transcoding based on low-complexity background modeling and adaptive block classification. IEEE Trans. Multimedia 15 (8), 1769–1785.

Zhang, X., Tian, Y., Huang, T., Dong, S., Gao, W., 2014. Optimizing the hierarchical prediction and coding in HEVC for surveillance and conference videos with background modeling. IEEE Trans. Image Processing 23 (10), 4511–4526.

Zhao, L., Tian, Y., Huang, T., 2014. Background-foreground division based search for motion estimation in surveillance video coding. Proceedings of IEEE Conference on Multimedia and Expo. pp. 1–6.

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

Zupancic, I., Blasi, S.G., Peixoto, E., Izquierdo, E., 2016. Inter-prediction optimizations for video coding using adaptive coding unit visiting order. IEEE Trans. Multimedia 1.