Real-time motion detection in H.264 compressed domain for surveillance application

Young Keun Kim, Yeong Gyoo Jeon and Seung Ho Shin*
ICT R&D Center, SK Telecom, Seongnam-si, Gyeonggi-do, Korea

*E-mail: shin5693@sk.com

Abstract. Camera motion detection is a useful functionality in video surveillance and many other areas, but with its widespread application in many areas, the compaction power has increased and needs to be addressed how to reduce it. So far, there has been a lot of research on pixel domain, but it is hard to operate with real-time and consumes very much computing power. However, since the compressed domain of block-based codec contains useful information for video analysis, this allows accurate video analysis at a low cost. The proposed method shows how to perform a motion detection accurately while significantly reducing the amount of computation compared to the pixel domain by using the DCT coefficients, motion vectors and other features in the compressed domain of the H.264 video. The proposed algorithm is designed for motion detection in video surveillance servers in the form of a cloud. The results of the experiment were shown to support validity.

1. Introduction
Video analysis is an important part of computer vision, but related research has been slower than image analysis, and most of studies has been done in the pixel domain. The pixel domain has abundant raw data, so it can be analysed for various applications, and most recently, the research using it is active due to the influence of deep learning. However, the analysis in the pixel domain consumes a lot of computational power to decode for video processing as a complexity of codec increases, such as high efficiency video coding (HEVC), and in addition, video analysis in the pixel domain is difficult to operate in real-time and requires more expensive cpu power equipment. As the codec advances, the resolution used in the video stream increases like full-high definition (FHD) and ultra-high definition (UHD). As the size increases, the throughput in the pixel domain increases accordingly. Therefore, video analysis that is suitable for the compressed domain video stream rather than analysis of raw pixel domain image is necessary.

Motion detection is widely used as a step toward picking out meaningful scenes from the video stream. However, this is often done in the pixel domain and a lot of computation power is used inevitably.

Video processing in the compressed domain of block-based codec is a method for accurate motion detection while overcoming computing power consumption. Studies of motion detection in the compressed domain have been keeping, but it still needs a lot of research. When the compressed domain is used, various features are extracted through entropy decoding using context adaptive variable length coding (CAVLC) or context adaptive binary arithmetic coding (CABAC), an initial partial process of the whole decoding process. Main features used in motion detection in the
compressed domain are motion vectors and discrete cosine transform (DCT) coefficients and most of studies has been done on motion vectors.

Motion vectors are generated by motion estimation during encoding, which has a wide diversity of motion estimation algorithms, and motion vectors generated through a motion estimation algorithm represent motions for encoding rather than motions that actually occurred. Therefore, there are always a number of motion vectors that are not accurate. In particular, there are many wrong motion vectors in the part where there is no pattern in the image or a certain pattern is repeated.

DCT coefficients are generated during the discrete cosine transform process that transforms the pixel domain into the frequency domain for each DCT coefficient block in the encoding process. The upper left side of each block is a low frequency region, and the lower right side is a high frequency region. DCT coefficients can also be useful for motion detection as it contains a lot of information needed to decompress during the decoding process. However, this does not happen for every block. It does not occur in Intra-coded blocks, but in inter-coded blocks. The DCT coefficients are divided into luminance (luma) and chrominance (chroma) components. The chroma component is divided into Cb coefficients and Cr coefficients, and each coefficient component is generated separately for one picture. As a result, three DCT coefficients occur. Where the DCT coefficient is high, a probability of motion existence is very high.

Motion vectors and DCT coefficients have distinctive features and complementary features, so accurate results are achieved if these features are combined effectively. Since motion detection in the compressed domain is given in macro block units, the accuracy is lower than that of the pixel domain, but sufficient information can be obtained as a pre-processing step for additional work.

It is important that motion detection used in the video surveillance system has high precision even if recall decreases. For example, in an application used in video surveillance that stores video stream each time a motion is detected, video stream without a motion frequently is stored if precision is not high, and this results in a large waste of storage space. If the recall rate is high, it would be more helpful to increase the motion detection accuracy, but it doesn't matter much if the motion detection algorithm doesn’t miss the object at a certain level. In addition, the probability of missing the motion of the object is reduced since video processing is mostly processed in group of pictures (GOP) units to perform temporal processing on multiple pictures.

This paper shows how to effectively perform motion detection to suit video surveillance using features in the compressed domain. The paper consists of the following. Section 2 summarizes the study on the compressed domain. Section 3 describes the method proposed. Section 4 shows an experiment on the performance of algorithms, and Section 5 summarizes the conclusions.

2. Previous works

Motion detection has been consistently studied with much interest as the first step in video analysis such as video surveillance and human-machine interaction. Generally, when making a video analysis, if the picture is selected through motion detection, performs additional tasks or analyse the pixel level after decoding it. Therefore, accurate motion detection is particularly important for the analysis of video streams. Much of research on motion detection was done in the pixel domain, but continuous studies have been executed in the compressed domain.

The first study of motion detection in the mpeg-2 video compressed domain was conducted by Mitsumoto et al. [1]. In this study, similar motion vectors are merged and noisy motion vectors are removed by using the magnitude and the direction of motion vectors. An assessment of motion vectors was performed to track the adjacent frames. Then DCT coefficients of the previous and current clusters are compared to determine whether the object actually moved or not. Eng et al. [2] used DCT coefficients for temporal analysis and a motion vector for spatial analysis. Jamrozik et al. [3] used leveled water shading to merge various areas based on motion vectors. Babu et al. [4] devised a method to detect a number of moving objects in a scene. This included em algorithm for segmentation. Zeng et al. [5] used DCT coefficients in DC image without using motion vectors. By doing so, they tried to overcome the noise of the motion vectors. Wang et al. [6] tried to obtain a segmentation of the
pixel level accuracy by using only DCT coefficients.

With the spread of H.264 following the mpeg-2 video, the studies on this have been continued steadily. Zeng et al. [7] used motion vectors of the H.264 compressed video for object segmentation. Moving objects were detected through the markov random field (MRF) classification process. Different motion vector types require use of different segmentation processes. Poppe et al. [8] performed motion detection using a macro block size and DCT coefficients without using motion vectors. This allowed them to process very quickly but their motion detection accuracy was significantly reduced. Mak et al, In the study of [9], MRF was used to model the foregrounds.

By using MRF, spatial and temporal continuities were preserved. However, this approach was only valid for large objects. Tom et al. [10] separated the foreground object using the size of the quantification parameter gradient information and motion vectors, unlike other algorithms.

Existing studies have detected motion using features such as motion vectors, DCT coefficients, a macro block size, etc. However, most of them focused on raising both recall and precision ratio, and the consideration of fast movements was insufficient. This paper proposes a motion detection method suitable for the video surveillance system in the form of cloud computing.

3. Proposed method

The approximate flow of a proposed algorithm is as in figure 1. As shown in figure 1, first, multiple features are extracted through entropy decoding process from the compressed domain to analyse each feature. Next, combination of features is executed. After that, a genetic algorithm is applied, then a gaussian filtering is performed, and the segmentation is finally performed before completing the motion detection. The proposed algorithm is intended to be used in the cloud-type applications in practical real-life situations, so it is designed for motion detection while increasing precision even at the expense of recall. For this purpose, luma DCT coefficients are used as the most important feature among many features. Video streams are composed of I, P, or B pictures. Since an I picture performs intra prediction on the entire region, not residual, in this case, a motion vector does not occur, and DCT coefficients appear irregularly, so an I picture is skipped due to difficulty in analysis. A detailed description of each step for motion detection is given in the following subsections.

Figure 1. Proposed motion detection method.
3.1. Feature extraction

The feature extraction step is responsible for extracting and analysing different types of features. Features used in this case are luma DCT coefficients, chroma DCT coefficients, motion vectors and macro blocks. The characteristics of each feature extracted are described below.

DCT coefficients occur when converting a pixel domain into a frequency domain during the discrete cosine transform process and occur in block units. This occurs for the luma component and the chroma component respectively and the chroma component is divided back into Cb and Cr components. This results in DCT coefficients for three components. In H.264 the DCT is performed in 4x4 or 8x8 units, most of H.264 use 4x4 block. In this paper, it is assumed that the 4x4 block unit DCT is performed. Performing the 4x4 block unit DCT produces the 4x4 DCT coefficient block as a result. Figure 2 shows the DCT coefficient block. The upper left of the block is a low frequency area, and the lower left is a high frequency area. It is not necessary to use all value when analysing DCT coefficients. Most of the values that are actually generated belongs to low frequency, so high frequency components are ignored. In addition, the DC area at the top left represents an area of little change and is ignored because it has little effect on motion detection. In this way, it is possible to reduce the calculation amount by limiting the range of the DCT coefficients used for motion detection. The actual area used for our method is marked as blue in figure 2.

| Luminance | Chroma Cb | Chroma Cr |
|-----------|-----------|-----------|

Figure 2. DCT coefficient blocks with usage selection.

Luma DCT coefficients increase greatly with slight movement, especially in the areas with distinct edges. However, the motion of the blurry section often does not show much change. The value of pixels selected is used as it is, otherwise the values is zero as Equation (1), where \( P_{\text{luma}}(x,y) \) is the luma DCT coefficient value corresponding to the \((x,y)\) coordinates for each luma DCT coefficient block. All the absolute values of the selected pixels in the block are added since the DCT coefficient can have both negative and positive values, as shown in Equation (2), where width is the width of the luma DCT coefficient block and height is the height of the luma DCT coefficient block and the result of the operation is \( A_{\text{luma}} \). If this value is large, the movement can be considered to have occurred, so it is used as a major feature for motion detection. If \( A_{\text{luma}} \) exceeds preconfigured luma single threshold \( (A_{\text{luma single th}}) \) as Equation (3), the block is determined that motion has occurred.

\[
P_{\text{luma}}(x,y) = \begin{cases} 
P_{\text{luma}}(x,y) & \text{if pixel is selected} \\
0 & \text{if pixel is not selected} 
\end{cases}
\]

\[
A_{\text{luma}} = \sum_{x=1}^{\text{width}} \sum_{y=1}^{\text{height}} |P_{\text{luma}}(x,y)|
\]

\[
R_{\text{luma}} = \begin{cases} 
\text{motion exist} & \text{if } (A_{\text{luma}} > A_{\text{luma single th}}) \\
\text{motion not exist} & \text{if } (A_{\text{luma}} \leq A_{\text{luma single th}})
\end{cases}
\]

Chroma DCT coefficients usually occur in the area where fast motion occurs. The color space used in most images is yuv 4:2:0, in which case the resolution of chroma DCT coefficients drops to 1/4 of the resolution of luma DCT coefficients. The absolute values of the Cb and Cr components in the
block is used as shown in Equation (4) ~ (6). Here, \( P_{cb}(x,y) \) and \( P_{cr}(x,y) \) are the dct coefficient values corresponding to the coordinates \((x,y)\) of the Cb DCT coefficient block and the Cr DCT coefficient block, respectively. And \( A_{chroma} \) is the result of the computation of chroma components, and if \( A_{chroma} \) exceeds \( A_{chroma,th} \) that is a preconfigured threshold, as in Equation (7), the block is determined that motion occurred. \( R_{chroma} \) is the result of motion detection for the chroma components. \( A_{chroma} \) shows less variance than \( A_{luma} \). The frequency of motion occurrence about \( R_{chroma} \) occurs less than that of \( R_{luma} \).

\[
P_{cb}(x,y) = \begin{cases} P_{cb}(x,y) & \text{if pixel is selected} \\ 0 & \text{if pixel is not selected} \end{cases} \tag{4}
\]

\[
P_{cr}(x,y) = \begin{cases} P_{cr}(x,y) & \text{if pixel is selected} \\ 0 & \text{if pixel is not selected} \end{cases} \tag{5}
\]

\[
A_{chroma} = \sum_{x=1}^{\text{width}} \sum_{y=1}^{\text{height}} |P_{cb}(x,y)| + |P_{cr}(x,y)| \tag{6}
\]

\[
R_{chroma} = \begin{cases} \text{motion exist} & \text{if } (A_{chroma} > A_{chroma,th}) \\ \text{motion not exist} & \text{if } (A_{chroma} \leq A_{chroma,th}) \end{cases} \tag{7}
\]

Motion vectors occur through motion estimation during encoding process. Motion vectors generated from motion estimation algorithm can vary depending on which algorithm is used because there are many different methods and motion vectors are generated to be advantageous for compression and are different from actual motion, and it usually occurs in the areas with no edge pattern or the areas where a constant pattern repeats. It occurs in sub macro blocks in Macro blocks. If it is intra prediction blocks, motion vector does not occur. Motion vectors that occur in large quantities are useful for motion detection, but it is very difficult to use motion vectors alone because there are many incorrectly occurring motion vectors. Therefore, using it with other features helps to increase accuracy. Since moving objects tend to move in the same direction, if the motion vector direction of the previous picture is similar to the direction of the current picture and the magnitude is more than a certain level, there is a high possibility of movement. In the case of noise, the direction and the magnitude of the motion vectors between the current location and the previous location severely changes, so noise can be removed by using the characteristics above.

Macro blocks have some constant size depending on movement. Macro blocks vary in size, such as 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4 and there is a tendency for smaller sub-macro blocks to occur when there is a movement. It is difficult to use the macro block size alone for motion detection like motion vectors, which occur without regard to actual motion. However, even if noise occurs, the size below 8x8 does not occur frequently, so this characteristic can be applied.

\subsection{Feature combination}

\begin{figure}[htb]
\centering
\includegraphics[width=0.5\textwidth]{fig3.png}
\caption{Feature combination of luma DCT coefficient and motion vector.}
\end{figure}
In the feature combination phase, each of the analysed features is combined. Luma DCT coefficients and motion vectors are the objects to combine. Luma DCT coefficients are correlated with motion vectors. The place where the motion vector occurs is the place where the most similar block is found, so the DCT coefficient value is smaller than the block without the motion vector. Therefore, if the motion vector and the DCT coefficient value are above a certain size respectively, it is considered that a motion occurs even if the value of a DCT coefficient is small. Since both the luma DCT coefficient and motion vector contain noise, each threshold is set for them to remove noise. Figure 3 shows how to determine the results of the motion detection by combination of luma DCT coefficients and motion vectors. Here, \( A_{\text{luma,comb,th}} \) and \( A_{\text{mv,comb,th}} \) are the threshold for the preconfigured luma DCT coefficient and the motion vector magnitude respectively and \( R_{\text{comb}} \) is the result of a feature combination.

### 3.3. Genetic algorithm

A genetic algorithm is used as an additional method for motion detection. By using motion vectors and macro blocks as features, the undetected part by the DCT coefficient-centred method described above can be supplemented. It works especially when there is a large movement.

If the initial population is selected incorrectly, the precision of motion detection is reduced, so it is selected through difficult conditions by mixing several features. The macro block size and the motion vector direction and the magnitude often contain significant noise, thus, it is necessary to prevent the selection of erroneous initial population.

The initial population is set to the areas that satisfy the following three conditions. First, two or more blocks with a sub macro block size of 8x8 or less exist consecutively. Second, the current and the previous motion vector direction occurring in the block is constant. Third, the magnitude is more than a certain size. Afterward, if the macro blocks of 8x8 or less are continuously located around the initial population using the characteristics of the macro block described above, the selection occurs for them to increase the detected area. When a selection occurs, crossover operation is occurred to a child block about magnitude and direction of the parent block. Then the selection and the crossover are repeated until there is no macro block size of 8x8 or smaller. If no corresponding block exists, a genetic algorithm shuts down.

### 3.4. Gaussian filtering and segmentation

Previous processes have tried to reduce noise while performing motion detection, but nevertheless some noise often occurs. This is the factor that decreases the precision, so gaussian filtering is performed to remove it. As a result, among the blocks that are determined to have motion, blocks are not continuous, and occurred apart from other blocks are filtered. Figure 4 shows the area where motion is detected after Gaussian filtering. The area is marked in orange.

![Figure 4. Detected motion after gaussian filtering.](image)

After applying gaussian filtering, segmentation is performed to distinguish the detected area for easy additional work in the next step, and then motion detection of the corresponding picture is
finished.

Noise has the characteristics of momentary occurrence and disappearance in a narrow area, so it can be detected and removed by temporal analysis for several consecutive pictures.

4. Experimental results

Several experiments were carried out to validate the proposed algorithm. The computer hardware specifications used for performance verification are 3.3 GHz Intel Core i5 CPU and 24 GB RAM.

First, the processing performance of an algorithm is measured. The frames per second (fps) is tested for 720P video and 1080P video. The fps is measured when the CPU usage for one core was 100% without synchronizing the video stream to timestamp. The results are shown at table 1 as follows.

| Resolution   | Frames per second |
|--------------|-------------------|
| 1280 x 720  | 188               |
| 1920 x 1080 | 132               |

Second, the motion detection performance of proposed algorithm at the object level and the picture level is measured. It is measured with precision and recall. Precision is calculated by Equation (8) and recall is calculated by Equation (9).

\[
\text{Precision} = \frac{\text{TruePositives}}{\text{TruePositives} + \text{FalsePositives}} \tag{8}
\]

\[
\text{Recall} = \frac{\text{TruePositives}}{\text{TruePositives} + \text{FalseNegatives}} \tag{9}
\]

Here is a short description of the parameters used. TruePositives - Number of correctly classified motion detected objects or pictures. FalsePositives - Number of stationary areas or pictures wrongly detected as moving object or picture. FalseNegatives – Number of moving objects or pictures wrongly detected as stationary area or picture.

The experimental process is carried out on 4 test videos shown as figure 5, which have different characteristics each other. Highway is the video encoded to H.264 in main profile with dataset of the IEEE Change Detection Workshop 2014 [11] as input. FFmpeg [12] is used to create compressed video from given pictures with all of encoding parameters set to default. And other sequences like Lobby, Mart, WallClock are the videos encoded to H.264 in main profile by surveillance cameras in different environments. The detected moving areas are overlapped in orange over the original image.

Due to the limitation of an algorithm that does not detect the precise movement by using macro blocks in the compressed domain, the performance experiments are performed at the picture level and the object level. The picture level experiment measures the motion detection accuracy of the picture unit for motions in the randomly selected picture. The object level experiment measures for motion detection accuracy of each object in a randomly selected picture of each test video.

![Figure 5](a) Highway, (b) Lobby, (c) Mart, (d) WallClock.

The performance test result is shown in figure 6~7. In measuring with the object unit, the average of precision is 86% and the average of recall is 76% and the standard deviation is calculated as 3% and 11% for precision and recall, respectively. In measuring with the picture unit, the average of precision is 97% and the average of recall is 87% and the standard deviation is calculated as 2% and 12% for precision and recall, respectively. Precision and recall vary depending on the characteristics of videos,
but tend to maintain high precision. There is also less variation in precision. Recall has a relatively large variation. Especially the precision in picture level is maintained above 94% and the standard deviation between videos is very low. These characteristics are expected to trigger services such as motion alerts, event recording, intrusion detection and so on in video surveillance systems that require accurate detecting in motion.

Figure 6. The performance test result for object units.

Figure 7. The performance test result for picture units.

5. Conclusions
In this paper, a method for detecting moving objects using the features of the H.264 compressed domain is proposed. This enables relatively accurate motion detection while consuming a little computing power than the pixel domain and also enables the high degree of precision, even though the percentage of recall is low. The proposed method is suitable for application in real-life situations like video surveillance servers in the form of cloud computing.

We plan to enhance the proposed algorithm and expand its usage by applying the proposed method to adjust the structure of the HEVC (high efficiency video coding) video stream in the future.

References
[1] S. Mitsumoto, H. Yuasa, and H. Zen. “Moving Object Detection from MPEG Coded Picture”, Proc. IAPR Workshop on machine vision and application (MVA'98), pp.422-25, (1998).
[2] H. L. Eng, K. K. Ma. “Spatiotemporal segmentation of moving video objects over MPEG compressed domain”, IEEE international conference on multimedia and expo, 3, pp 1531–34, (2000).
[3] M. Jamrozik, M. Hayes. “A compressed domain video object segmentation system”, International conference on image processing, 1, pp 113–16, (2002).
[4] R. V. Babu, K. R. Ramakrishnan, S. H. Srinivasan. “Video object segmentation: a compressed domain approach”, IEEE Transaction on circuits and systems for video technology, 14, pp 462–74, (2004).
[5] W. Zeng, W. Gao, D. Zhao. “Automatic moving object extraction in MPEG video”, Proceedings of the international symposium on circuits and systems, 2, pp 524–27, (2003).
[6] W. Wang, L. Yang, W. Gao. “Modeling background and segmenting moving objects from compressed video”, IEEE Transaction on circuits and systems for video technology, 18, pp 670–81, (2008).
[7] W. Zeng, J. Du, W. Gao, Q. Huang. “Robust moving object segmentation on H.264/AVC compressed video using the block-based MRF model”, Real-Time Imaging , 11, pp 290–299, (2005).
[8] C. Poppe, S. D. Bruyne, T. Paridaens, P. Lambert, R. V. de Walle. “Moving object detection in the H.264/AVC compressed domain for video surveillance applications”, Journal of visual
communication and image representation, 20, pp 428–37, (2009).
[9] C. M. Mak, W. K. Cham. “Real-time video object segmentation in H.264 compressed domain”, IET Image Process, 3, pp 272–85, (2009).
[10] M. Tom, R. V. Babu. “Fast moving-object detection in H.264/AVC compressed domain for video surveillance”, National conference on computer vision, pattern recognition, image processing and graphics, (2013).
[11] 2nd IEEE change Detection workshop, in conjunction with CVPR 2013, www.changedetection.net, (2014).
[12] FFmpeg, http://www.ffmpeg.org.