Fast compressed domain motion detection in H.264 video streams for video surveillance applications

Szczerba, Krzysztof; Forchhammer, Søren; Støttrup-Andersen, Jesper; Eybye, Peder Tanderup

Published in:
Proceedings, AVSS

Link to article, DOI:
10.1109/AVSS.2009.78

Publication date:
2009

Document Version
Publisher's PDF, also known as Version of record

Fast compressed domain motion detection in H.264 video streams for video surveillance applications

Krzysztof Szczerba, Søren Forchhammer
Technical University of Denmark
DTU Fotonik
Ørsteds Plads b.343
DK-2800 Kgs. Lyngby
krsz@fotonik.dtu.dk, sofo@fotonik.dtu.dk

Jesper Støttrup-Andersen, Peder Tanderup Eybye
Milestone Systems A/S
Banemarksvej 50G
DK-2605 Brøndby
Denmark
jsa@milestone.dk, pey@milestone.dk

***Abstract*—This paper presents a novel approach to fast motion detection in H.264/MPEG-4 Advanced Video Coding (AVC) compressed video streams for IP video surveillance systems. The goal is to develop algorithms which may be useful in a real-life industrial perspective by facilitating the processing of large numbers of video streams on a single server. The focus of the work is on using the information in coded video streams to reduce the computational complexity and memory requirements, which translates into reduced hardware requirements and costs. The devised algorithm detects and segments activity based on motion vectors embedded in the video stream without requiring a full decoding and reconstruction of video frames. To improve the robustness to noise, a confidence measure based on temporal and spatial clues is introduced to increase the probability of correct detection. The algorithm was tested on indoor surveillance H.264 sequences.**

I. Introduction

IP video surveillance systems are growing in size, complexity and capacity. Introduction of advanced coding standards like the H.264/MPEG4-AVC [8] means that better compression is achieved, but also that more resources have to be allocated to decoding of the video. Higher resolution images delivered by newer cameras require more processing time for motion detection and segmentation algorithms based on background subtraction or frame-to-frame-change detection. Performance of pixel based processing in large systems may be challenged by the load of decoding multiple streams and processing of vast amounts of video.

Motion detection is often the very first step of the analysis of the video, used either for triggering alarms or to determine which video sequences have to be stored. This task is performed continuously on all video streams in the system. In such a set-up, the motion detection and extraction must be fast and accurate, to avoid omission of important parts of the video, which may later be used in a legal case. The accuracy and sensitivity must be good enough to detect the events but not to trigger many false alarms.

A solution for reduction of the amount of computing power required is to use the data from the compressed video. MPEG-derived video coding standards exploit the temporal correlation between frames. In simple terms, each block has a motion vector (MV) attached to it, describing it's displacement from a reference frame to the current frame. The MVs are roughly corresponding to the optical flow. It must be stressed however, that some MVs may be placed in the video because of coding gain, and not because they reflect true motion. Therefore they MVs extracted should be evaluated.

In this paper, we present a method for reliable motion detection using the MVs from the coded video stream. A low complexity evaluation of the extracted MVs is performed to assign confidence in relation to true motion. Extraction and manipulation of the MVs is itself a much easier task than full reconstruction of each pixel of a video frame. The MVs describe motion of group of pixels, called blocks, which also contributes to reduction of computational complexity. The goal is a fast algorithm for enabling processing of large numbers of video streams on a single server.

This paper is organized in the following way. In Section II related work on motion detection is discussed. The proposed algorithm is presented in Section III. In Section IV the test methodology is discussed, and results are presented in Section V.

II. Motion detection in surveillance

The majority of work on change detection and motion detection for surveillance has naturally been performed in the pixel domain based on reconstructed images. Only a minor part of the papers have been devoted to the possibilities of doing this in a compressed domain.

A. Pixel domain approaches

A comprehensive survey of change detection algorithms in pixel domain was provided by Radke [7]. This survey presents a wide range of change detection algorithms along with methods for pre-processing of input images and post-processing of the resultant change masks. The survey covers a wide area of applications of change detection, including video surveillance. Another recent study of pixel domain change detection is provided by Parks and Fels [6], where the focus was on background subtraction algorithms.

978-0-7695-3718-4/09 $25.00 © 2009 IEEE
DOI 10.1109/AVSS.2009.78

B. Compressed domain approaches

Compressed domain approaches are new and less explored compared to pixel domain processing. An example of a simple and fast algorithm for compressed domain change detection is presented by Bracamonte [3]. Their algorithm works by comparing the phases of the DCT (Discrete Cosine Transform) coefficients in a series of images. The presented results are good, however this algorithm only works for MJPEG and key frames in MPEG derived video coding standards.

Another approach to compressed domain information is to rely on the motion vectors (MVs) in the code stream. Babu [2] presented an algorithm for extraction of independently moving objects, using MVs from an MPEG-4 video stream. Their focus was on object segmentation rather than speed, and they interpolated a dense motion field as part of the processing. Similar work was done by Zeng [12], where they presented a robust method for moving object segmentation from H.264/AVC coded video streams aiming at real time applications. They presented very good results on object segmentation, however the speed reported, ranging from 300-800 ms per frame (CIF resolution), is not high enough for our objective of a large video surveillance system working with a large number of cameras (tens per standard PC class server). Their algorithm, using Markov Random Fields, employs iterative processing. Another recent work [11] both addressed localization and action recognition. The method is fast but still slower than our goal and the test sequences were persons on simple backgrounds. In an earlier paper [10] a simpler approach to compressed domain motion extraction and detection was presented, however the results provided are preliminary in nature.

C. Comparison of compressed domain and pixel domain approaches

Both approaches to video processing have their advantages and disadvantages. Change detection or motion detection on reconstructed images is conceptually appealing and in simple implementations can be fast, usually at some cost of reliability. The methods most commonly applied in video surveillance systems seem to be the background subtraction methods based on various statistical background models. The work by Hampapur [1] is an example of activities in searching surveillance video using background subtraction. Pixel domain approaches are independent of the video coding standard used and there is a great potential for very reliable detection using advanced pre- and post-processing. Although this method can be made reliable it has a serious disadvantage in a video surveillance system where all the video is stored and transmitted in a compressed format, and where video reconstruction might handicap the system performance. Reconstruction of the necessary video might be a daunting task, especially in case where all video streams are checked for motion. Compressed domain processing avoids the full decoding and reconstruction of the video, which provides a potential for real time processing of multiple video streams on one server. Compressed domain processing also has the advantage of extracting video stream data, which has been generated using the original non-compressed data, which will not be available when processing a decoded stream. Thus the lossy video coding introduces a noise component, that will have to be dealt with.

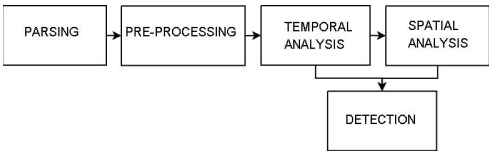

Figure 1. Block diagram of compressed domain motion detection.

III. Fast compressed domain processing

The general approach of the proposed motion detection method is to extract and analyze the motion vectors of the video stream. The overview is presented in Figure 1. The first step is parsing the video stream to extract the MVs for each frame. The MVs are then pre-processed before applying temporal and spatial analysis. The goal of the analysis is to find the motion vectors which reflect real motion. On the basis of this analysis the detection of motion is performed.

Intuitively, it may seem that (code stream) MVs instantly provide perfect description of ongoing motion. This, is unfortunately not the case, because the MVs in MPEG-derived video coding standards, as H.264, are used to improve compression, rather than to provide motion description. MVs not related to motion can occur due to wrong match of regularly textured areas, noise, low-light conditions etc.

Another problem is that there are intra coded blocks (I-blocks), which do not have motion vectors, as opposed to inter coded blocks (P-blocks). The I-blocks are often inserted when a new object enters a frame and at occlusions. In general the coder will select intra coding if this yields less bits (relative to the distortion) than coding a P-block with a motion vector. A detailed discussion of these issues is complex, but the important thing to note is that there may be I-blocks without motion vectors among P-blocks.

A. Parsing

Parsing is the process of extraction of the MVs (and position of I-blocks) from the coded video stream. The parsing process requires entropy decoding and reconstruction of MVs, but this process is a minor effort compared to complete decoding and reconstruction. The H.264 coding standard allows for the use of tree structured motion compensation [8], i.e. the 16×16 macro blocks (MB) may be partitioned into smaller blocks having side lengths of 4 and 8 pixels as well as 16. We simplify the handling of block sizes by

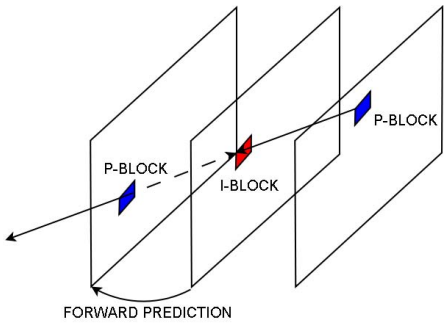

Figure 2. Interpolation of MVs for I blocks

mapping the extracted vectors to a single block size - 4×4, in all cases. In case the partition is bigger than 4×4, all 4×4 blocks within the partition are assigned the same MV. The MVs lengths are processed at quarter-pixel accuracy, since H.264 allows for quarter pixel precision motion estimation.

B. Pre-processing

To remedy the problem of missing MVs at I-blocks, the extracted motion field is pre-processed prior to analysis. The basic assumption is that motion of the real moving objects of interest should be smooth and of relatively uniform speed in surveillance applications. The missing MVs in a current frame are interpolated from MVs from the previous and the next frame as illustrated in Figure 2. The dashed line represents inversion of an MV in order to point to the next frame, instead of the previous one. The MV for the I-block is interpolated as an average of the MV from the past frame and an MV from the next frame pointing to this I-block. The interpolated motion vector is incorporated into the motion field, thus providing a complete set of MVs over the 4×4 blocks. In the rest of the paper MV refers to values from this complete set.

C. Confidence measure

The complete set of motion vectors can be further analyzed to estimate which MVs represent real motion. Real motion of interest should be smooth and continuous in time and space in surveillance applications. The determination of which MVs correspond to true motion is based on spatial and temporal clues. An overview block diagram is presented in Figure 1.

Each MV resulting from the pre-processing is checked if it represents real motion. A measure, of how likely a motion vector is to represent a real motion, is introduced and called the confidence. The *confidence* can be thought of as a (Bayesian) probability that a given MV is related to a real moving object and is a number in the range $(0, 1)$. It is calculated based on temporal and spatial relations of MVs. A similar confidence measure was introduced in the paper by Wang [10]. It is reasonable to assume that all physical moving objects have certain inertia and are not able to abruptly change direction of motion or velocity, therefore MVs corresponding to abrupt change of motion are assigned lower confidence. True motion will produce MVs which are consistent over time. The confidence is calculated for all non-zero MVs in the frame and saved in an array corresponding to the number of 4×4 blocks in the frames. This associates a confidence with each block, but also with the motion vector of the block. Therefore, the terms "confidence of motion vector" and "confidence of block" are used interchangeably and they refer to the same variable. The confidence will by definition only be nonzero for blocks with nonzero length of MV. Thus the array with confidence values describes the likelihood of having a moving object in a given part of the frame.

D. Confidence calculation

The confidence is evaluated both temporally and spatially. Temporal confidence is based on correlation of temporally adjacent MVs and spatial confidence is based on spatial clustering of (high) confidence motion vectors. The confidence measures from temporal and spatial aspects are averaged to give the final confidence number as detailed below.

1) Temporal confidence: For the temporal confidence process we first define a *reference MV)* (RMV) for each MV in the currently processed frame. The RMV is assigned the value of the motion vector of the 4×4 block containing the center of the area pointed to by the MV of the current block. The block from which the MV was taken is called the reference block. To access the temporal confidence, we calculate the following values: The length of an MV (LMV), the difference of lengths between the current MV and RMV ($DLMV$), and the difference in angle between the current MV and (RMV), ($DAMV$). To evaluate we also consider the type of the reference block (RT), i.e. is it intra or non-intra. Further let CR denote the confidence value of an RMV and CMV denote the temporal confidence of the motion vector in the current frame. All of these variables are defined at block level. The notation will both be used to refer to the full array for a frame and the individual value of a given block. The variables are initially compared to the following thresholds,

1) T_{LMV} - lower threshold for LMV
2) T_{DMV} - upper threshold for $DLMV$
3) T_{DAMV} - upper threshold for $DAMV$

which provides the basic part of the temporal confidence algorithm as shown in Algorithm 1. The algorithm calculates a confidence value CMV for the MV of each 4×4 block in a frame based on the RMV and their corresponding confidence, CR.

2) Spatial confidence: The temporal confidence values, CMV, for the current frame are saved in an array and processed spatially using two subsequent morphological operations - closing and opening. Use of morphological

Algorithm 1 Temporal confidence estimation in a single frame.

Input: MV, RMV, CR
Parameters: T_{LMV}, T_{DLMV}, T_{DAMV}
Initialize first frame with $CR = 0$
for all 4×4 blocks **do**
  Calculate $LMV, DLMV, DAMV, CR$
  if $RT \neq intrablock$ **then**
    if $LMV > T_{LMV}$ **and** $DLMV < T_{DLMV}$ **and** $DAMV < T_{DAMV}$ **then**
      $CMV = 0.5 + CR \times 0.5$
    else
      if $MV = (0, 0)$ **then**
        $CMV = 0$
      else
        $CMV = CR \times 0.5$
      end if
    end if
  else
    $CMV = 0.5$
  end if
end for
return CMV

closing and opening for postprocessing of motion masks is also described e.g. in [7]. The goal of the morphological processing is to increase the uniformity of the confidence map. Values of the spatial confidence obtained from the morphological filtering are averaged with the values from the temporal confidence process defining a *spatio-temporal confidence* measure. This reduces the confidence of elements removed by the spatial filtering, however it still leaves a trace of them.

3) Final motion detection: The final step is creating a binary motion mask to extract the detected motion based on the spatio-temporal confidence array. A confidence threshold value T_C is selected, and 4×4 blocks with spatio-temporal confidence above T_C are marked as detected motion blocks. The number of these blocks in each frame is found and used to detect motion at the frame level. In our set-up, motion is detected when at least one block is above T_C.

4) Settings adjustability to noise: The thresholds used in the algorithm can be adjusted to tune the algorithm towards noise resistance or greater sensitivity. Increasing the threshold for minimal MV length and final confidence threshold increases the noise robustness, but increases chances for omission of little or irregular motion. The parameters in Algorithm 1 may be adjusted accordingly. In the test we select $T_{DLMV} = 7.5$ pixels and $T_{DAMV} = 90$. To adjust sensitivity we select $T_{LMV} = 0.75$ pixels and $T_C = 0.5$ for good light conditions and $T_{LMV} = 1.5$ pixels and $T_C = 0.85$ for robustness in low light conditions. The settings may potentially be adjusted based on meta-data, application specific or by automatic noise level estimation, eg. based on DCT-coefficients of the I-frames.

Table I
VIDEO TEST SEQUENCE SETTINGS

Resolution	704×576	320×240
Frame rate	25fps	25fps
GOP length	25	25
Rate control	constant & variable	constant
Range of bit rates	287-2158 kbps	260-1632 kbps
Light conditions	varied	varied

IV. TESTS

Indoor sequences were recorded using an IP based surveillance system to test the algorithms. The sequences were recorded with H.264 capable cameras from two different vendors, Sony (320×240 pixels) and Axis Communications (704×576 pixels). A wide range of settings of video encoding was used to test how the algorithm works in various scenarios. Special attention was paid to use a wide range of bitrate settings, as the bitrate setting is crucial for encoder decisions.

A. Test data set

All 23 sequences were recorded in H.264 Baseline profile, without B-frames, a GOP (Group of Picture) length of 25 frames and one reference frame only. The range of settings is presented in Table I. The sequences were 500-800 frames long. The frame rate was targeted at 25 fps, however in video sequences recorded with rate control it varied slightly. Sequences were recorded indoor with artificial light and low ambient light, as a first setting, and low ambient light only as a second setting of light to stress the algorithm. Likewise test sequences with a small object (distant person) moving in a corridor behind a glass window was also included.

B. Effects of level of light

The reduced level of light, in the low light sequences, triggered the camera sensors to increase sensitivity. This gave a higher noise level in the video, resulting in turn in a larger number of "noisy" MVs in the coded video. These sequences would also be challenging to a pixel domain algorithm, because the noise is originally induced in the pixel domain.

C. Test methodology

Evaluation of the motion detection algorithm can be done visually or qualitatively by comparison with a ground truth set. Visual evaluation is easy only in principle, because it requires many observers to get a conclusive result and it is time consuming. Ground truth is also challenging, because coming up with a good ground truth is difficult. This problem is well explained in a paper by Hu [5].

Figure 3. Motion masks, big moving object in the scene

Figure 4. Motion masks, distant object moving in corridor behind window glass.

The motion was evaluated at frame level. This means that if a frame contained any activity it was marked as a frame with motion and vice versa. The ground truth for motion detection was labelled with three levels - no motion, ambiguous motion and certain motion. The label *ambiguous motion* was introduced to account for the beginning or end of motion, objects partially hidden etc. Detection of motion in a frame while the frame was marked in the ground truth as having motion was evaluated as a true positive. Detection of motion in a frame, while the frame was marked in ground truth as not having motion was evaluated as false positive. The percentage of true positives (TP) over a video stream is calculated as

$$TP = N_{tp}/N_m$$

where N_{tp} is the number of true positive frames and N_m denotes the number of frames labelled as motion in the ground truth. The percentage of false positives (FP) over a video stream is calculated as

$$FP = N_{fp}/N_{nm}$$

where N_{fp} is the number of false positive frames and N_{nm} is the amount of frames without motion in the ground truth.

The false negatives are 100% minus the percentage of true positives. By analogy the true negative are 100% minus the percentage of false positives.

It must be stressed, that for event detection it is not necessary to detect all the individual frames which have motion, because of temporal continuity of events in surveillance video.

V. Test results

The test set-up reflects typical video surveillance for indoor monitoring. Results on the test sequences showed good performance in terms of motion detection and reliability. All motion *events* in the 23 test video streams were detected. The extracted masks of the moving object did vary with settings of the algorithm parameters. The results on the test sequences showed robustness towards the variety of scene settings, sizes of moving object, and object distances. An example of detection of a big object is presented in Figure 3. Smaller objects were also detected, as shown in Figure 4. The masks of the moving objects are however affected by the noise robustness settings as also illustrated by Figures 3 and 4. The high noise robustness setting worked well with noisy sequences recorded in dark rooms, but the motion masks produced were less consistent. If needed, the quality of the motion masks can be improved by morphological operations or clustering. The main focus in our work was on motion detection, whereas improving the motion mask was less important.

It was observed that there was a dependency on bitrate for the amount of false positive detections in noisy video sequences with rate control. The higher the bitrate, the more false positives there were. This is however not a problem in video sequences recorded at normal light conditions. It was observed that image resolution did not seem to have impact on detection performance. (The moving objects were larger than the 2-4 macro blocks which would seem to be the limit of the current version.) The most influential parameter was the light intensity. Low light intensity means poorer signal-to-noise-ratio in the camera CCD sensor, because shot noise and dark current noise become significant in relation to signal power. The noise sources in CCD cameras are discussed in [4] and in [9], which discusses the statistical calibration of CCD imaging. Low-light intensity resulted in significant camera noise and produced a significant number of noisy MVs.

A. Noise robustness

Table II
AVERAGE PERCENTAGE OF FALSE AND TRUE POSITIVES FRAMES IN THE TEST VIDEO SEQUENCES.

	False positives	False positives, high noise robustness	True positives	True positives, high noise robustness
Normal light	1.4	0.0	98.3	86.2
Low light	57.2	2.3	99.9	98.2

Thorough testing and comparison against ground truth was conducted on the 23 sequences. Of these there were 3 noisy and 3 non-noisy sequences at 320×240 and 5 noisy and 12 non-noisy sequences at 704×576. Table II presents the average test results for both high and low noise robustness settings, with distinction for sequences recorded in normal and low light intensity. The results show that for the low noise robustness setting and low light intensity there is a large percentage of false positives. The problem is remedied by the high noise robustness setting. Detection at frame level of motion in normal light intensity was reduced

by the high noise robustness settings. However no motion event was missed in this setting.

B. Comments on algorithm

Based on the prototype implementation, we assess that an efficient implementation of the algorithms should facilitate running the motion detection on tens of H.264 video streams on a single server. The algorithms were implemented and tested with H.264/MPEG AVC streams, but they can be ported to other hybrid video coding standards, which use block-based motion compensation. Support for multiple reference frames in H.264 streams could also be supported, using scaling of the MVs according to the temporal distance they span.

VI. Conclusions

The presented compressed domain method provides a simple and fast solution to motion detection in video surveillance systems. A major advantage of the algorithm is that it does not require full reconstruction of the video, which reduces the amounts of data being processed and the overall processing time. By parsing the coded stream, motion vectors and position of intra blocks are extracted. The motion vectors are processed and analyzed using fast non-iterative techniques. Incorporating a confidence measure and adjustability of the algorithm helps adaption to various (indoors) environments. Tests on indoor (H.264 coded) test sequences showed that the motion events were detected, as well as the majority number of individual frames within the motion events.

References

[1] A. Hampapur Smart Video Surveillance for Proactive Security. *IEEE Signal Proc. Mag.*, 131–134, July, 2008.

[2] R. V. Babu, K. R. Ramakrishnan, and S. H. Srinivasan. Video object segmentation: A compressed domain approach. *IEEE Trans. Circ. Syst. Video Tech.*, 14(4):462–474, 2004.

[3] J. Bracamonte, F. P. M. Ansorge, and P.-A. Farine. A low complexity change detection algorithm operating in the compressed domain. *Proc. 8th COST 276 Workshop, Information, Knowledge Management, Integrated Media Communications*, 7(12), 2005.

[4] G. Healey and R. Kondepudy. Radiometric ccd camera calibration and noise estimation. *IEEE Trans. Pattern Anal. Machine Intel.*, 16(3):267–276, 1994.

[5] J. Hu, , R. Kashi, D. Lopresti, G. Nagy, and G. Wilfong. Why table ground-truthing is hard. In *Proc. 6th Int'l. Conf. Document Anal. Recognition*.

[6] D. H. Parks, , and S. S. Fels. Evaluation of background subtraction algorithms with post-processing.

[7] R. J. Radke, , S. Andra, O. Al-Kofahi, and B. Roysam. Image change detection algorithms: A systematic survey. *IEEE Trans. Image Proc.*, 14(3):294–306, 2005.

[8] I. Richardson. *H.264 and MPEG-4 Video Compression*. John Wiley & Sons, September 2003.

[9] Y. Tsin, V. Ramesh, and T. Kanade. Statistical calibration of ccd imaging process. *Proc. 8th IEEE Int'l Conf. Computer Vision, ICCV 2001*, 1:480–487 vol.1, 2001.

[10] R. Wang, , H.-J. Zhang, and Y.-Q. Zhang. A confidence based moving object extraction system built for compressed domain. *IEEE Int'l. Symp. Circuits Systems*, 2000.

[11] C. Yeo, P. Ahammad, K. Ramachandran, and S. S. Sastry. High-speed action recognition and localization in compressed domain videos. *IEEE Trans. Circ. Video Tech.*, 18(8), 2008.

[12] W. Zeng, , J. Du, W. Gao, and Q. Huang. Robust moving object segmentation on h.264/avc compressed video using the block-based mrf model. ELSEVIER, Real-Time Imaging 11 (2005) 290299.