EAI Endorsed Transactions
on Industrial Networks and Intelligent Systems
Research Article EAI.EU

Reducing Bitrate and Increasing the Quality of Inter Frame by Avoiding Quantization Errors in Stationary Blocks

Xuan-Tu Tran[1,*], Ngoc-Sinh Nguyen[1], Duy-Hieu Bui[1], Minh-Trien Pham[1], Hung K. Nguyen[1], and Cong-Kha Pham[2]

[1]VNU Key Laboratory for Smart Integrated Systems (SISLAB), VNU University of Engineering and Technology, Vietnam National University, Hanoi (VNU) – 144 Xuan Thuy road, Cau Giay district, Hanoi 123106, Vietnam.
[2]The University of Electro-Communications (UEC) – 1-5-1 Chofugaoka, Chofu, Tokyo 182-8585, Japan.

Abstract

In image compression and video coding, quantization error helps to reduce the amount of information of the high frequency components. However, in temporal prediction the quantization error contributes its value as noise in the total residual information. Therefore, the residual signal of the inter-picture prediction is greater than the expected one and always differs zero value even input video contains only homogeneous frames. In this paper, we reveal negative effects of quantization errors in inter prediction and propose a video encoding scheme which is able to avoid side effects of quantization errors in the stationary parts. We propose to implement a motion detection algorithm as the first stage of video encoding to separate the video into two parts: motion and static. The motion information allows us to force residual data of non-changed part to zero and keep the residual signal of motion regularly. Beside, we design block-based filters which improve motion results and filter those results fit into block encode size well. Fixed residual data of static information permits us to pre-calculate its quantized coefficient and create a bypass encoding path for it. Experimental results with the JPEG compression (MJPEG-DPCM) showed that the proposed method produces lower bitrate than the conventional MJPEG-DPCM at the same quantization parameter and a lower computational complexity.

Received on 13 September 2019; accepted on 11 December 2019; published on 17 January 2020

Keywords: Video compression, video coding, motion detection, MJPEG

doi:10.4108/eai.24-10-2019.162795

1. Introduction

The growth of Internet of Things (IoT) in industry has been garnering substantial momentum in the last few years. As Gartner's prediction [9], the number of internet-connected IoT devices (not including smart phones and personal computers) will reach 20 billions by 2020. One important subset of the IoT devices is Internet-enabled surveillance cameras (i.e., Internet Protocol cameras or IP cameras). According to MarketsandMarkets' report, the value of video surveillance applications market achieved $25.5 billion in 2016 and is expected to reach $71.28 billion by 2020 [6]. IP cameras are well sold for consumer and business uses. Everything from keeping an eye on the kids, the nanny and the wildlife in the garden, through to monitoring shops, offices and car parks will be possible.

The rapid emergence of embedded video surveillance favorably opens a chance of developing a specific video coding for this area. In contrast of encoding movie videos, embedded surveillance camera systems demand a video codec that keeps running at very low power consumption, low resource requirement as much as possible and has an ability to do pre-processing, but responses real-time signals.

Various standards have been used for compression digital video signal. They can be classified into two categories: Motion Estimation based approaches such as H.264/AVC (Advanced Video Coding), HEVC (High Efficiency Video Coding); and still image compression

*Corresponding author: Xuan-Tu Tran. Email: tutx@vnu.edu.vn

approaches such as Motion JPEG (MJPEG), Motion JPEG2000 (MJPEG 2000). Recently, almost surveillance systems support H.264/AVC or MJPEG, or even both of them.

MJPEG is known as the simplest video codec [3]. It encodes each frame of video sequence separately into a JPEG bitstream. Compared to other video codecs, MJPEG requires a minimum hardware resource and is strongly recommended for limited systems. Another advantage is that it is only based on JPEG standard, thus it is widely supported and can be easily setup. However, it produces encoded bitstreams which are much higher bitrate than H.264/AVC or H.265/HEVC bitstreams at the same quality. An encoded MJPEG bitstream of video at $640 \times 480@30fps$ consumes 5 to $10Mbps$. Therefore, we have to encode the difference between two continuous frames as in[10] to reduce frame rate and chose a suitable frame resolution to adapt the available bandwidth and storage capability.

H.264/AVC is known as MPEG-4 Part 10/AVC for Advanced Video Coding. It was firstly introduced in 2003 [4]. H.264/AVC provides good video quality at substantially lower bitrate than previous standards without increasing the complexity of design so much. H.264/AVC encoder can achieve average image quality with a compression ratio of $60:1$, while MJPEG's compression ratio is only up to $15:1$, and previous MPEG-4 standard is only a half compression ratio as H.264/AVC at a similar quality. Therefore, H.264/AVC has rapidly replaced previous MPEG standards for movie applications which accept very high latency. Implementing H.264/AVC in real-time application is hardly at all because of its complexity and latency. An embedded camera surveillance system using H.264/AVC requires a high performance center processing unit or a system with hardware codec.

H.265/HEVC is the successor of H.264/AVC video codec standard. It achieves twice compression ratio as much as H.264/AVC [5] but requires more power full system to be implemented. HEVC standard is designed for high resolution video, up to 8192×4320 (8K Ultra-High-Definition (UHD)). At this moment, embedding HEVC into surveillance camera systems is quite limited because of the hardware constraints.

Each discussed video codec has its advantages and disadvantages. However, they share the same video coding scheme Hybrid based on Differential Pulse Code Modulation (DPCM), and none of them considers side effects of quantization error and any characteristics of surveillance video sequences. Quantization error, which is the difference between original data and de-quantized data, plays as the key to reduce the amount of information of high frequency components that are low sensitive to human eyes system. However, in inter prediction, quantization error prevents detecting stationary blocks and lets the residual information differ zero value even there is no change in series frames. Including quantization errors in residual information of static part not only decreases the compression ratio and the quality of the inter frame but also increases the Entropy coding time. Then, for each stationary block, if whose residual information is zero, we will save the bitrate and keep the quality as its predictor without any quantization factor dependency. In surveillance, there is a large percentage of non-motion area which lets us achieve good exchange.

In this work, we reveal side effects of quantization error on inter frames and propose a video encoding diagram for surveillance applications which is able to eliminate the negative effects of quantization errors in inter frames. The proposed video coding has ability to recognize the change between frames by implementing a motion detection algorithm at the first stage. Beside that, block based filters are also designed to improve the quality of motion detection algorithm and filter motion results fit into block encode size. The class division lets create a bypass encoding path to avoid quantization errors and speed up coding process for static areas.

The remaining part of this paper is organized as follows. Section 2 reveals side effects of quantization error on inter frame. Section 3 provides a solution for quantization error issues. Section 4 introduces a fast and robust motion detection algorithm, called Zipfian, which plays as motion detection in the proposed video coding scheme. Section 5 presents a block-based filter for motion detection. Section 6 shows the detail of the proposed video coding scheme and Section 7 is the results of test case with the JPEG standard. Finally, some conclusions and future works will be given in Section 8.

2. Effect of Quantization Error in Inter Frame

In video codecs, to reduce transmission rate of digital picture information and guarantee encoder to share the same set of reference pictures with its decoder, the Differential Pulse Code Module (DPCM) has been applied. However, the use of DPCM requires some cautions, the frame transmission error tends to propagate and severely degrade the frame/video quality.

Figure 1 shows the basic block functions of DPCM in video coding. In this figure, each input block I, the transmission error/residual information R is the difference between the current block data and its reference data I', $R = I - I'_{t_1}$. The energy of residual data R is significant less than the raw data I, but it is still more than the expectation $R = I - I_{t_1}$, especially inter frame. This problem comes from the quantization error $Qe = Q' - Q$ in the reference data/decoded buffer image/reconstructed buffer image $I'_t = I'_{t_1} + R' = I'_{t_1} + inverse\_transform(Q + Qe)$. Beside increment

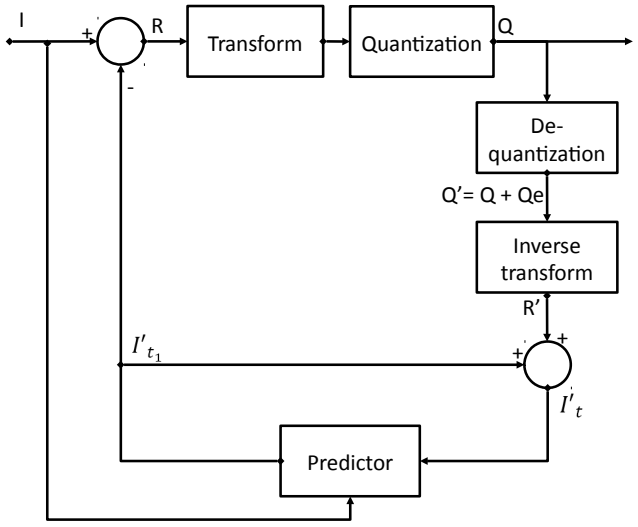


Figure 1. Differential pulse code modulation.

transmission error energy, the quantization error also degrades the video quality at inter frames.

A typical case in surveillance video has been simulated to reveal side effects of quantization at inter-frames as in Figure 2. In this simulation, a video contains only homogeneous frames, and from the second frame are encoded by two methodologies: DPCM model and fixed R to the expected value. In the DPCM scheme, the error R_2 is the difference between I_2 and the decoded data D_1 of I_1 JPEG bitstream. Another methodology, the residual data R_{exp} is the difference between the twin frames and $R_{exp} \equiv 0$. Residual data from both methodologies is encoded to JPEG bitstream. Then, we compare result both methodologies at two sides: size of encoded bitstream and the quality of reconstructed frame.

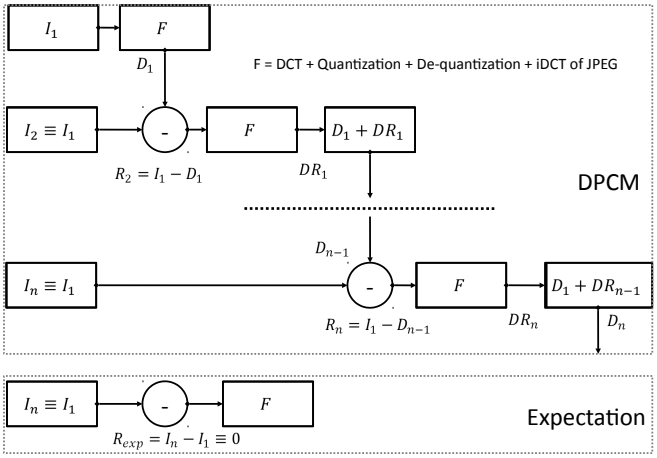


Figure 2. Simulation model of effects of quantization error on inter frame.

Figure 3 shows the simulation results of two methodologies with generated videos, which are made of only the first frame in Hall and Akiyo sequences. The left side of Figure 3 is the diagram of residual values R_2 at Quantization Parameter (QP) =75 at the second frame in generated video. As shown, the R_2 values has cosines shape, the peak value is up to more than 20/255 (255 is number values of 8-bit pixel data) and the average of absolute $|R_2(x)|$ is more than 1.4. Consequently, in the right side of Figure 3, the bitstream size of all inter frames from DPCM model are always greater than the expectation, and the difference increases by each QP. In range $QP <$ 20, the difference bitstream size is nearly zero because of very high quantization coefficients. QP in range [20:70], it increases slightly, and at $QP = 70$, the size of inter bitstream from DPCM is twice as much as the expectation. From $QP = 70$, the difference increases significantly by each QP. Although using DPCM increases bitstream size, however the reconstructed frame quality still degrades and the quality is inverse proportion to the size. Because, if the R_2 increases, the error between $D_2 = DR_2 + D_1$ and D_1 will increase. As shown, we lose both the quality and bitrate of the inter frame.

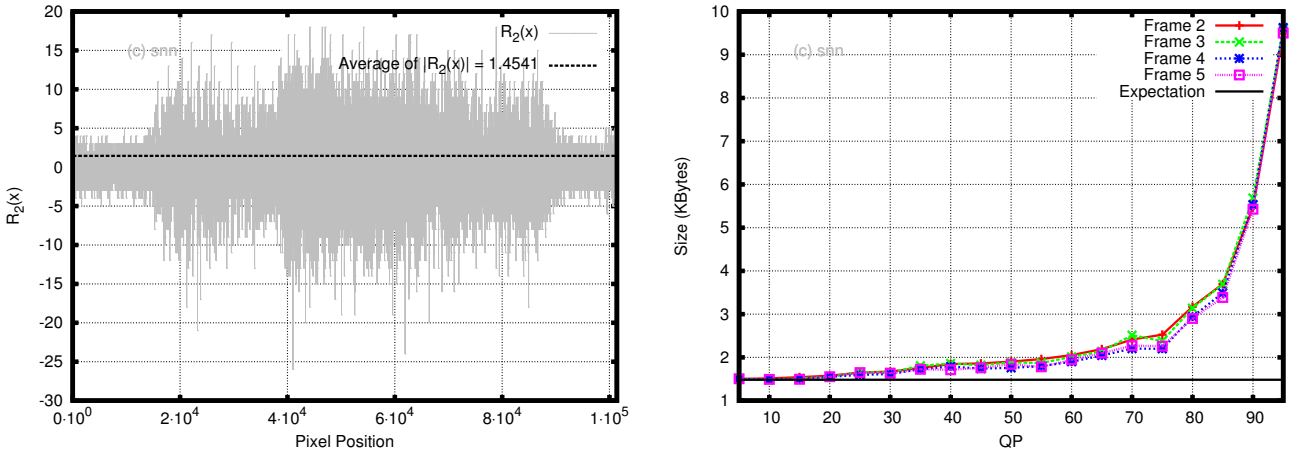
Another problem of DPCM or quantization error in inter prediction is that it provides untrusted information of non-motion frame. In video coding, the Group of Picture (GOP) structure has been applied since MPEG-1. This parameter guarantees that almost all of scenes in current encoding frame I and its predictor are similar. However, fixed the GOP parameter wastes the bitrate for encoding frame by intra-prediction when there is no change in series frames. Dynamic (adaptive) GOP is one of the solutions, but quantization error always provides us non-zero residual frame data (motion frame). Hence, applying dynamic GOP requires a more trusted information of non-motion frame.

3. Avoiding Quantization Error in Static

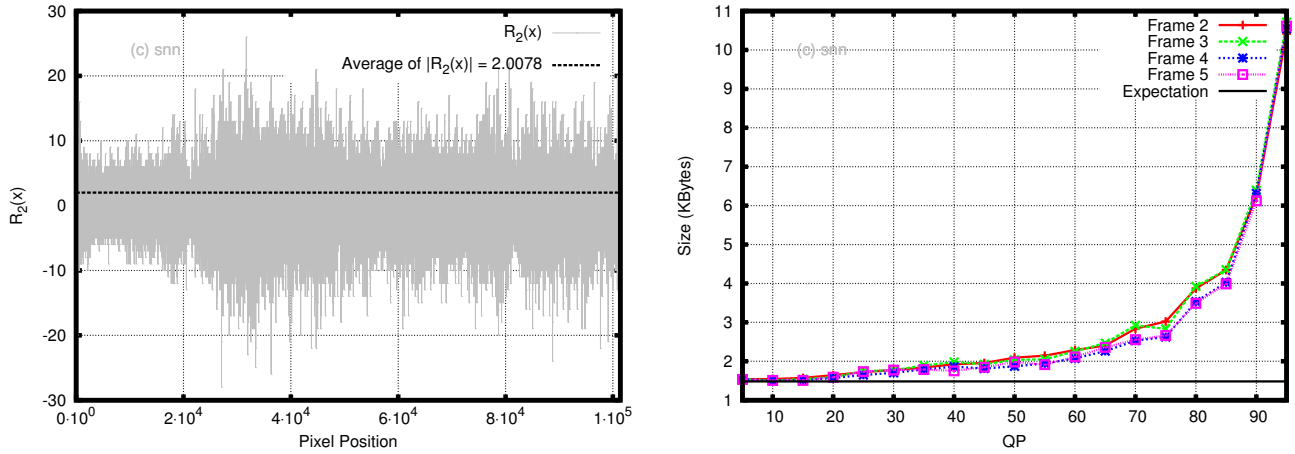
The couple quantization and de-quantization processes produce quantization errors by themselves, except only one situation that $DCT(R_{m\times n}) \equiv 0_{m\times n}$. Hence, a solution for those cases as the simulation in Figure 2 is forcing the error $R_{m\times n} = 0_{m\times n}$ when all pixels in a block are unchanged.

$$I_t(x) = I_{t'}(x) \Rightarrow \begin{cases} R(x) = 0 \\ D_{t'}(x) = D_t(x) \end{cases}$$

Ideally, detecting unchanged pixels is done easily by computing the difference between adjacent frames. However, this methodology does not filter invisible objects in human eyes systems such as windy, slight



(a) Video generated from the 1st frame of Akiyo sequence.



(b) Video generated from the 1st frame of Hall sequence.

Figure 3. Residual error (left side) and bitstream size (right side) results of the test case in Figure 2.

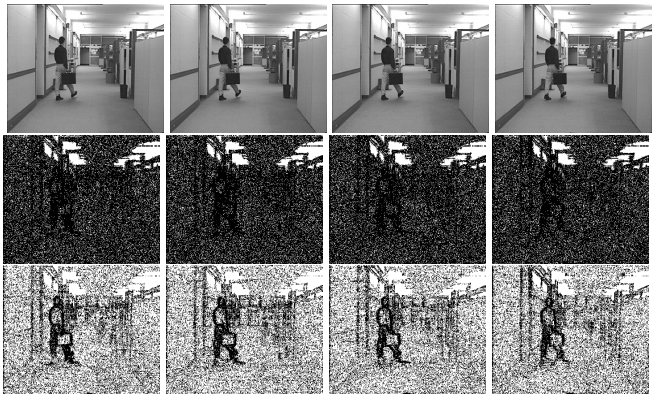
lightning, etc or unstable stage of camera. Additionally, DPCM also contributes its quantization errors as noises into the prediction data. Figure 4 shows the results of detecting unchanged pixels by differential methodology $I_t(x) - I_{t-1}(x)$ (without quantization error factor) of two video sequences, Akiyo and Hall. At the first row, almost all of frames are static in human eyes systems. It takes more than 70% of frame area. Perfectly, the mask of pixel status should be all white with only black around and inside the human. However, this methodology without threshold as in the second row of Figure 4a and Figure 4b presents the masks of motion and stationary pixels horribly. Because the recorded frame is under the influence of the unstable stage of image sensor and includes invisible motions in human eyes systems. The difference with threshold provides the mask is much better than without threshold. However, choosing a threshold value affects to the accuracy of system directly.

The works in [7, 11] describe the powerful of motion detection algorithms. They are widely used in surveillance, monitoring systems and alert whenever a strange motion appeared. Basically, motion detection algorithms have ability of focusing on interesting motion objects and filtering unwanted motions. Those features open an opportunity of implementing motion detection algorithms into video codec to force the residual information of stationary part to zero value $R(x) = 0$.

4. A Simple Motion Detection Algorithm: Zipfian

The growing interest of full automatic surveillances promotes the development of motion detection algorithms. By the time, more and more motion detection algorithms are invented [7, 8, 11–14]. All of them try to separate all pixels of each frame into two classes: the background - pixels belong to static scene; and the foreground - pixels belong to moving scene.

(a) Akiyo sequence.



(b) Hall sequence.

Figure 4. Result of the differential methodology.

Background subtraction is a widely used technique for detecting moving objects in static cameras. This technique tries to build a model of background. Then, the foreground model is obtained by comparing the difference between current frame and the background model. Recently, Sigma-Delta, Zipfian and Vibe are the most attractive algorithms use this technique [7, 11–13].

Zipfian Estimation is a well-known low complexity motion detection algorithm [12], which is a combination of the Sigma-Delta algorithm with a statistical estimation, called a Zipf-Mandelbrot distribution. Algorithm 1 shows all steps in Zipfian Estimation. Firstly, the variance threshold σ according to the frame indexed t is computed: $\sigma = \frac{2^m}{2^p}$, where m is the bit length of a pixel, p is the greatest 2^p value that divides (t mod 2^m). The current background M_t is updated from previous background model M_{t-1} whenever variance V_t is greater than the variance threshold σ. Next, O is the absolute difference between current input frame I_t and the current background model M_t. To avoid auto-reference, the variance V_t is updated by a period Tv and the previous variance which is not equal $N \times O$. Where, Tv is a variance update period and N is an amplification factor of the variance V_t (usually from 1 to 4). Finally, a pixel x is motion if only if its absolute difference $O(x)$ is over its variance $V_t(x)$.

Algorithm 1: Zipfian estimation

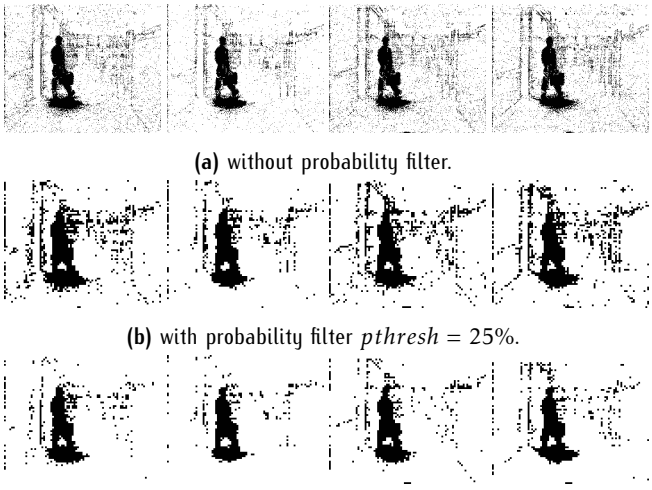
```
find the greatest 2^p that divides (t mod 2^m)
set σ = 2^m / 2^p
find updateV = t%T_v
foreach pixel x do
    if V_{t-1}(x) > σ then
        if M_{t-1}(x) < I_t(x) then M_t(x) ← M_{t-1} + 1 ;
        if M_{t-1}(x) > I_t(x) then M_t(x) ← M_{t-1} − 1 ;
foreach pixel x do
    O_t(x) = |M_t(x) − I_t(x)|
foreach pixel x do
    if updateV = 0 then
        if V_{t-1}(x) < N × O_t(x) then
            V_t(x) ← V_{t-1}(x) + 1 ;
        if V_{t-1}(x) > N × O_t(x) then
            V_t(x) ← V_{t-1}(x) − 1 ;
foreach pixel x do
    if O_t(x) < V_t(x) then E_t(x) ← 0;
    else E_t(x) ← 1;
```

The advantage of the Zipfian Estimation is straight forward to compute in any fixed-point arithmetic, using a limited instruction set: absolute difference, comparison and increment/decrement. Thus, it is well adapted to real-time applications. Compared to other motion detection algorithms as in [7], the Zipfian is the fastest algorithm; its speed is twice time as much as the second fastest (Vibe). The limitation of the Zipfian Estimation is the quality of the foreground, it still has many noises - static pixels, and extracting foreground can be inefficient in complicated background. Those noises prevent force residual block R in Section 3 to zero matrix.

5. Block based Filter for Motion Detection Algorithm

Motion detection algorithms are able to discard small motions. However, the mask of pixel status still has many noises, especially low cost algorithms like series Sigma-Delta. It prohibits forcing the residual data of invisible or unwanted motion R to zero matrix. Block based filters are designed to filter those noises and bring the output to nearly truth ground. The fundamental of filters are from the characteristic of results from motion detection algorithm and the real motions. Additionally, applying filters in block shape size fits with rectangular video frame and block based image transform such as DCT, Wavelet.

In fact, motion probability around human visible motions is much greater than noises or invisible



(a) without probability filter.

(b) with probability filter $pthresh = 25\%$.

(c) with probability filter $pthresh = 40\%$.

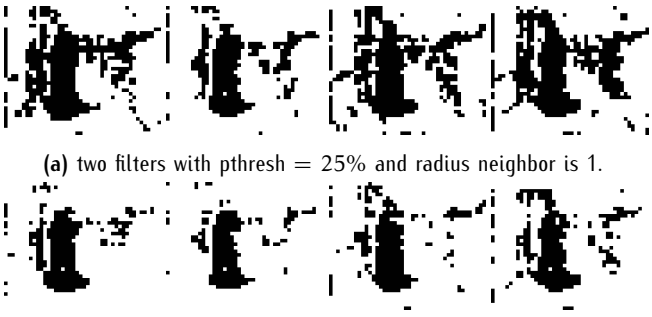
Figure 5. Zipfian Estimation without/with the probability filter.

motions. From this fact, analyzing the probability of motion in a block is a feasible solution. Algorithm 2 is an example of using probability methodology to classify each block into static or motion type. As shown, the sum of motion pixels *sum* in each block is computed and compared to a threshold, called block threshold *bthresh*. The *bthresh* represents the probability threshold *pthresh* in specific block size, $btresh = pthresh \times$ block size.

Algorithm 2: Probability filter

```
Data: Pixel status P(x)
Data: phtresh is motion probability threshold
Result: Block status B(x), 0 is static, 1 is motion
Initialization sum = 0
foreach pixel x in block do
    if E(x) = 1 then
        sum = sum + 1
        if sum ≥ pthres × blockSize then
            return B(x) = 1
return B(x) = 0
```

Figure 5 shows the output of Zipfian Estimation algorithm without and with the probability filter. In this test, the block size is 4×4 as the current minimum supported block size in video coding. As shown, the first row is the only Zipfian Estimation, the two other rows are Zipfian Estimation with probability filter $pthresh = 25\%$ and 40%, respectively. The two last rows with probability filter clearly give us much better results than without filter. Almost all of unwanted motions/noises in static scene are removed, and regions of true motions are focused. This result supports forcing the residual matrix to zero matrix as well as avoid



(a) two filters with pthresh = 25% and radius neighbor is 1.

(b) two filters with $pthresh = 40\%$ and radius neighbor is 1.

Figure 6. Zipfian Estimation with two proposed filters.

the quantization error of static scenes. The higher the *pthresh* is, the more small motion region is reassigned as stationary is.

Another fact is that the size of an interesting objects is equal to several blocks. And, whenever an object moves, the changed pixels include pixels at the previous object position and the current object position. As shown in Figure 5, there are several isolated motion blocks (note that a motion block does not have any motion neighbors), which maybe from unstable stage of camera or invisible motion. It makes senses to remove more redundancies from isolated motion block. Normally, detecting an isolated block is done via considering the characteristic of its neighbors. Moreover, to increase the accuracy of motion detection algorithm, static blocks around large motion or motion zone are reassigned to motions. This helps us keep detail view of both the motions and its environment.

Algorithm 3: Isolation filter

```
Set const = 2 is a threshold to reassign stationary to
  motion
Find sum is total number of motion neighbors foreach
  neighbor of block x do
    if B(x) = 1 then
        sum + 1
if sum > const | {x = 1 && sum > 0} then
    motion
else
    stationary
```

Algorithm 3 is an example of isolation filter algorithm. In this algorithm, *sum* is the total number of its motion neighbors. Whenever, the current block is motion and there is at least one motion neighbor, this block is true motion block. And, whenever the *sum* is greater than *const*, it means that the current block is around or inside a true motion zone, then this block will also be assigned as a motion block. In contrast, the current block will be stationary.

Figure 6 presents the results of Zipfian Estimation with two proposed filters. Low motion probability blocks are classified to stationary block, all of isolated motion blocks are reassigned to stationary and the motion regions are more focused.

In the isolation filter, the status of the current block depends on the status of its neighbors. Hence, the conventional processing order as in the JPEG, HEVC requires doing motion detection whole neighbors before. It prevents running motion detection, filters and encoding in parallel or using pipeline techniques. To support those tasks work parallel or pipeline, a new block processing order are used. Figure 7 shows the proposed processing order in sub-sampling Y only and YCbCr 4:2:0.

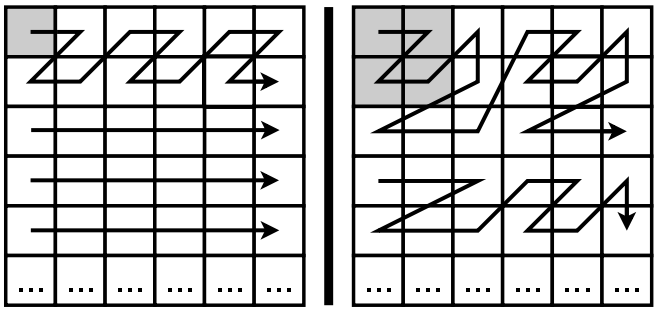


Figure 7. Zigzag block processing order [15].

6. Proposed Video Encoder Avoids Quantization Error of Static Scene in Inter Frame

In Section 2 and Section 3, we have discussed the negative influence of quantization error on static scenes and provided a feasible solution. This section, we would like to introduce the detail implementation of those experienced into video encoding scheme.

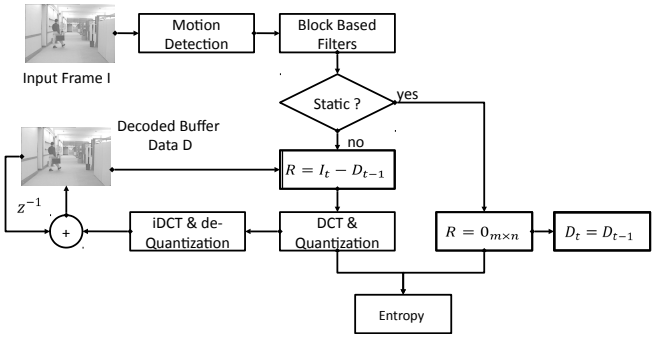


Figure 8. Proposed encoding architecture for fixed predictor from the previous decoded video frame.

Figure 8 shows the proposed encoding architecture for predictor from the previous decoded video frame. Normally, the residual data is the difference between current data and its predictor, and is computed firstly. However, to eliminate side effect of quantization errors, the residual computation depends on the block status. Hence, motion detection and block-based filters are processed firstly as in Figure 8. For a stationary block, its residual block is equal to zero matrix and its quantized block is also zero matrix absolutely without any computation. Other blocks, the residual information is the difference between current data and predictor data from previous decoded frame $R(x) = I_t(x) - D_{t-1(x)}$. Finally, residual data of both block types is encoded by the same Entropy encoding. In reconstruction stage, the decoded data D_t of a stationary block are assigned to previous decoded data block D_{t-1}. To reconstruct data of motion block, the quantized data has to go through de-quantization, inverse transform and reconstruction processes, respectively.

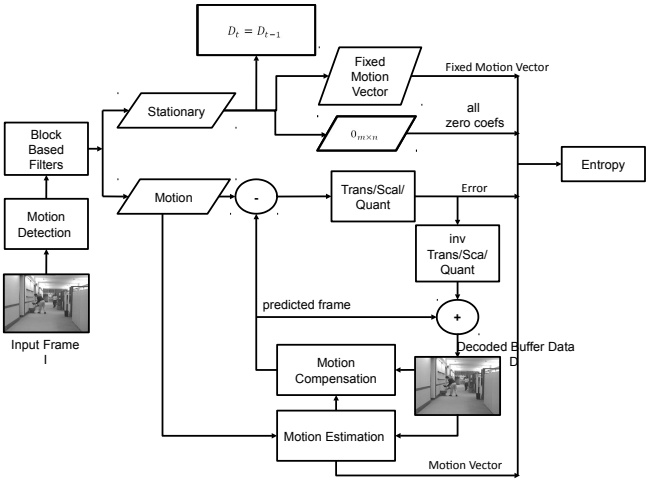


Figure 9. Proposed encoding architecture for Motion Estimation based video standards.

Figure 9 shows the proposed encoding architecture for Motion Estimation based standards such ad HEVC or H.264/AVC. Compared to Hybrid encoding architecture, this architecture has two more processes: motion detection and block-based filters, and a bypass encoding path for stationary block. The same as Figure 8, the two new processes in Figure 9 run firstly and defines status of each block. Encode data of stationary block includes matrix of zero and a fixed motion vector which points to the same block position in previous decoded buffer. Absolutely, the decoded data of stationary $D_t(x)$ is equal to previous decoded data $D_{t-1}(x)$. Others blocks are encoded and reconstructed step by step as in conventional Hybrid video coding scheme.

As described, those ideas have to pay cost for motion detection and block based filters processes, and save cost in encoding and reconstruction path of stationary blocks. Comparing to the conventional Hybrid (DPCM) video coding scheme, the coding time and bitrate of this

scheme depend on three more factors: the complexity of motion detection algorithm, block based filters, and static percentage.

7. Experience of the Proposed Video Encoding Architecture with JPEG Standard

Implementing JPEG image compression standard in the proposed video encoding architecture is chosen for number reasons. JPEG is really low complexity, is wide supported by the industry and used in low performance system. In JPEG, there are Discrete Cosine Transform (DCT) and quantization processes which are implemented in almost all of video coding standards. Additionally, JPEG has no standard and no GOP constraint.

In this test, the JPEG is implemented into the proposed encoding architecture for fixed predictor from previous decoded video frame as in Figure 8. Because JPEG bit-stream only accepts data in range [0:255]. Hence, the residual data R and reconstruction data D of motion are computed as in Equations 1-2, respectively.

$$R(x) = \frac{I_t(x) - D_{t-1}(x)}{2} \tag{1}$$

$$D_t(x) = 2 \times R'(x) + D_{t-1}(x) \tag{2}$$

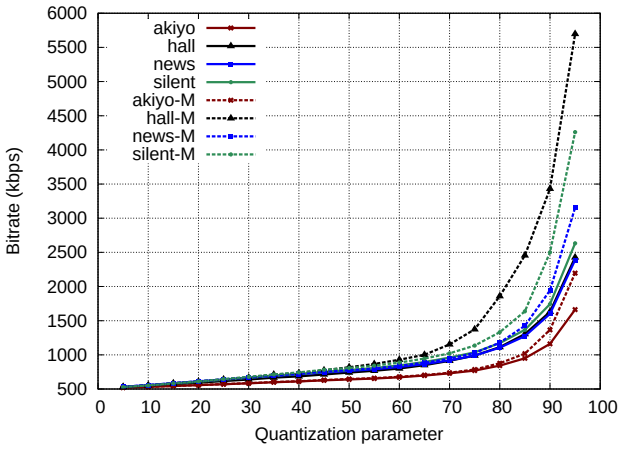


Figure 10. Bitrate comparison: JPEG with DPCM and JPEG in simulated model.

Figure 10 shows the bitrate of the JPEG with DPCM; and the JPEG with the proposed coding methodology. All lines with suffix "-M" means that the videos are encoded by JPEG with DPCM, otherwise, videos are encoded by JPEG in the proposed encoding architecture. The proposed methodology always provides lower bitrate than DPCM model. The difference bitrate goes up when increasing the quantization parameter QP. At small QP, the bitrate of two models are quite similar because of very high quantization coefficients. At QP in range [30:70], the difference goes slightly and linearly. However, from QP=70 (most used QP), it increases significantly by each QP. Figure 12 shows more details of the bitrate in Figure 10 at typical quantization parameter [50-80]. As shown, the saved birate is from several tens kbps to several hundreds kpbs. All the saved bitrate are from forcing residual data of stationary to zero value.

As the complexity dependency of proposed video codec in Section 6, Table 1 summarizes number operations of all main processes per block 8×8 in proposed encoding architecture: Zipfian Estimation, Block-Based filters, transform (DCT) and quantization The operations set includes comparison, shift, addition/subtraction, multiply and division. In Zipfian Estimation algorithm, all of its parameters: N and Tv are set to 4, and the total number of operations is the worst case. As shown in the table, to classify block status, preprocessing stage spends up to 472 operations, but more than $\frac{3}{4}$ all of operations is comparison. About the couple DCT & quantization or the couple inverse DCT & de-quantization, each one spends nearly 1000 operations per block 8×8. This number is twice as much as preprocessing stage. Moreover, almost all of operations in (inv) DCT transform & quantization is additions, multiplications, and the bit-length data in those operations is higher than the bit-length of pixel data. We have taken an example to examine the trade off between JPEG with the proposed encoding architecture and JPEG with DPCM at complexity side. Assuming that, all operation types have the same complexity level and an image has 10 blocks 8×8, p blocks are stationary, (inv) DCT & quantization spends 1000 operations per block, preprocessing stage spends 500 operations per block. Then, we have the total number of operations in JPEG with DPCM s_1 and JPEG with the proposed model s_2 as in Equations 3 - 4. In this comparison, the total number of operations in JPEG with proposed model is less than total number of operations in JPEG with DPCM whenever stationary probability p is greater than 25%.

$$s_1 = 2 \times 10 \times 1000 \tag{3}$$

$$s_2 = 2 \times (10 - p) \times 1000 + 10 * 500 \tag{4}$$

$$s_1 > s_2 <=> p > 2.5 \tag{5}$$

Figure 11 represents percentage of motion in some sequences: Akiyo, Hall, New and Silent. As shown, the peak motion percentage is only up to a half. At this percentage, implementing JPEG into the proposed architectures is able to eliminate side effects of quantization errors, while it still keeps the complexity at lower level than JPEG with DPCM.

Beside quatitative computing, we have simulated the model at Figure 8 in testing environment as in Table 2.

Table 1. Complexity comparison

Process	Comp.	Shift	Add/Sub	Mult	Div
Zipfian Estimation	312	16	144	0	0
Probability filter	76	0	12	0	0
Isolation filter	11	0	8	0	0
DCT	0	128	624	192	0
Quantization	0	0	0	0	64
inverse DCT	0	160	632	192	0
De-quantization	0	0	0	64	0

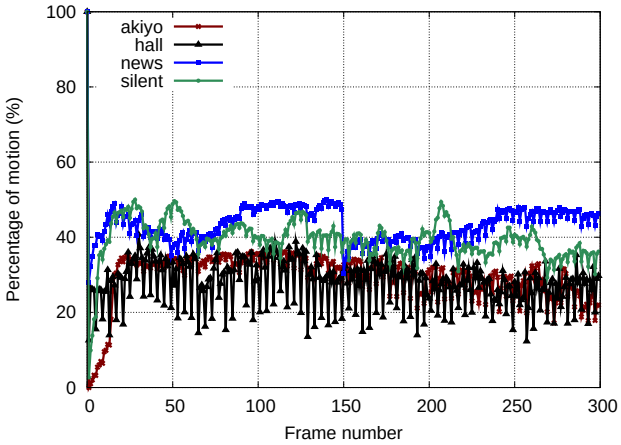


Figure 11. Percentage of motion.

Table 2. Testing environment

Environment	Name	Information
CPU	Intel E4400	Intel E4400
OS	Centos 6.5	Centos 6.5
Library	JPEG 6b	Stable version 6b[1]
Compiler	GCC	Enable -O2 flags
Block size in pre-processing stage	4×4	4×4
Block size in JPEG	8×8	8×8
Video	Y4M	CIF, YCbCr 4:2:0, 30fps [2]

Figure 13 represents coding time of JPEG with DPCM and JPEG in the proposed architecture. The coding time is achieved from "user" time of "time" command in Linux operating system. The running time at each quantization parameter is the average time of 100 run times. As shown, the proposed model always provides smaller coding time than JPEG with DPCM. At the same quantization parameter, the proposed model saves at least 15% coding time as JPEG with DPCM. The coding times of JPEG

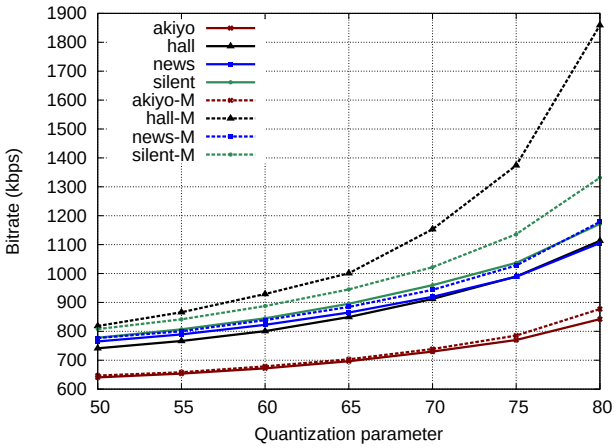


Figure 12. Bitrate comparison: JPEG with DPCM and JPEG in simulated model at QP [50-80].

with DPCM increases significantly when increasing the quantization parameter. In contrast, JPEG with the proposed model is almost independent with QP. Because only motion blocks joint to full encoding processes (included reconstruction). As quatitative computation in Equation 5, the smaller coding time understands well.

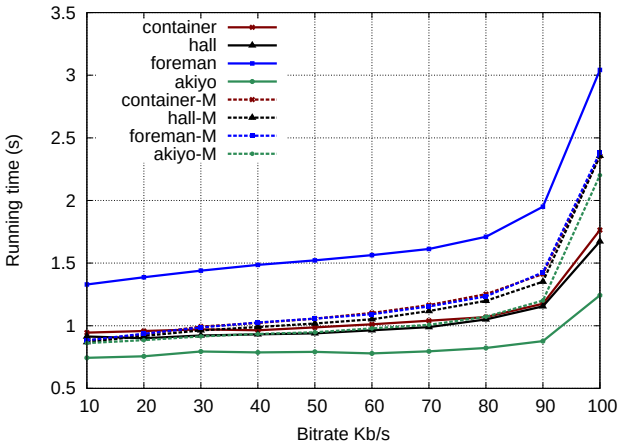


Figure 13. Coding-time comparison: JPEG with DPCM and JPEG in the proposed architecture.

8. Conclusion

The paper presented how the quantization error affects on stationary blocks. Beside increasing the encoded bitstream size and reducing the quality of reconstructing inter-frame, the entropy encoding time is also increased. To solve those problems, we proposed a novel encoding video scheme, which implements an efficient motion detection algorithm and block-based filters. Those additions help to separate motion and stationary parts, and provide necessary information to

creates bypass in encoding path for static. Experimental results with MJPEG-DPCM proved that encoding static by the proposed path reduces bitstream size, increases quality of reconstruction frame and reduces total coding time, especially high stationary percentage video. At the same quantization factor, the proposed encoding scheme saves bitrate from tens *kbps* to hundreds *kbps*, and the running time is less than 85% as running time by MJPEG-DPCM. Those results open an opportunity to embed this coding scheme into embedded surveillance camera systems.

Acknowledgment

This work has been supported by Vietnam National University, Hanoi(VNU) under the project number QG.18.38. The authors would like to thank anonymous reviewers for their helpful comments.

References

[1] Independent JPEG Group. http://www.ijg.org/.

[2] Xiph.org :: Derf's Test Media Collection. http://media.xiph.org/video/derf/.

[3] ITU-T Recommendation T.81:. Recommendation, 1992.

[4] Recommendation ITU-T H.264/AVC. Technical report, ITU-T, 2003.

[5] Recommendation ITU-T H.265/HEVC. Technical report, ITU-T, 2013.

[6] Video Surveillance Market by System (Analog, IP), Component (Camera, Monitor, Server, Storage Device, Software), Service (VSaaS, Maintenance), Application (Infrastructure, Commercial, Institutional, Defense, residential), and Geography - Global Forecast to 2022. Technical report, Marketsandmarkets, 2016.

[7] O. Barnich and M. Van Droogenbroeck. ViBe: A Universal Background Subtraction Algorithm for Video Sequences. *IEEE Transactions on Image Processing*, 20(6):1709–1724, June 2011.

[8] S. Denman, C. Fookes, and S. Sridharan. Improved Simultaneous Computation of Motion Detection and Optical Flow for Object Tracking. In *2009 Digital Image Computing: Techniques and Applications*, pages 175–182, December 2009.

[9] Mark Hung. Leading the IoT – gartner insights on how to lead in a connected world. Technical report, Gartner, 2017.

[10] S. kamath and J. R. Jackson. Low-bit rate motion JPEG using differential encoding. In *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004.*, volume 2, pages 1723–1726 Vol.2, November 2004.

[11] L. Lacassagne, A. Manzanera, and A. Dupret. Motion detection: Fast and robust algorithms for embedded systems. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 3265–3268, November 2009.

[12] Antoine Manzanera. Sigma-Delta Background Subtraction and the Zipf Law. In *Progress in Pattern Recognition, Image Analysis and Applications (CIARP'07)*, pages 32–51. Lecture Notes in Computer Science - Springer Verlag, 2007.

[13] Antoine Manzanera 1st AAntoine Manzanera and Julien C. Richefeu. A Robust and Computationally Efficient Motion Detection Algorithm Based on Sigma-Delta Background Estimation. (PDF Download Available). In *CVGIP 2004, Proceedings of the Fourth Indian Conference on Computer Vision, Graphics & Image Processing*, 2004.

[14] A. Mittal and N. Paragios. Motion-based background subtraction using adaptive kernel density estimation. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–302–II–309 Vol.2, June 2004.

[15] N. S. Nguyen, D. H. Bui, and X. T. Tran. Reducing temporal redundancy in MJPEG using Zipfian estimation techniques. In *2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pages 65–68, November 2014.