Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 421307, 20 pages
doi:10.1155/2012/421307

Research Article

Video Compression Schemes Using Edge Feature on Wireless Video Sensor Networks

Phat Nguyen Huu,[1] Vinh Tran-Quang,[2] and Takumi Miyoshi[3]

[1] *Graduate School of Engineering and Science, Shibaura Institute of Technology, Saitama 337–8570, Japan*
[2] *School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi 112400, Vietnam*
[3] *Department of Electronic Information Systems, College of Systems Engineering and Science, Shibaura Institute of Technology, Saitama 337-8570, Japan*

Correspondence should be addressed to Phat Nguyen Huu, m709506@shibaura-it.ac.jp

Received 4 May 2012; Revised 3 September 2012; Accepted 6 September 2012

Academic Editor: Chi Ko

This paper puts forward a low-complexity video compression algorithm that uses the edges of objects in the frames to estimate and compensate for motion. Based on the proposed algorithm, two schemes that balance energy consumption among nodes in a cluster on a wireless video sensor network (WVSN) are proposed. In these schemes, we divide the compression process into several small processing components, which are then distributed to multiple nodes along a path from a source node to a cluster head in a cluster. We conduct extensive computational simulations to examine the truth of our method and find that the proposed schemes not only balance energy consumption of sensor nodes by sharing of the processing tasks but also improve the quality of decoding video by using edges of objects in the frames.

1. Introduction

A wireless video sensor network (WVSN) is a special kind of wireless sensor network (WSN) that is capable of capturing video data at video sensor nodes (nodes have the equipment to capture video data), processing the data, and transferring them using a multihop technique to the base station. Two types of nodes are considered in WVSNs, video sensor nodes (source nodes) and processing sensor nodes that affect the retrieval of video data. The main functions of video sensor nodes are to capture objects and to record video data, while the main functions of processing sensor nodes are to gather data and to process them. The size of video data is large while all nodes are limited by their resources, that is, power battery, computational capacity, and memory. Therefore, saving energy consumption of network by reducing transmission data size and guaranteeing quality of service (QoS) are two fundamental issues in WVSNs.

With the practical deployment of WSNs and the availability of small CMOS camera chips, many multimedia applications have been studied on WVSN. These applications provide a distributed sensing and monitoring environment for the ability to retrieve video data, including surveillance, computer vision, video tracking, remote live video and control, and smart homes [1–3]. In these applications, researchers concentrate attention not only on observing something in static scenarios but also on discovering the changes. However, it is too difficult to implement both of these tasks on WVSNs because of limited energy and processor speed of sensors [4]. To solve these problems, we propose a new algorithm for compressing video data on WVSNs. In the algorithm, we use a homogeneity detection technique [5–9], which is based on the color edge detection algorithms to improve the quality and compression rate of decoding video.

There are several image edge detection algorithms such as Sobel, Gaussian, homogeneity methods [8–12]. In our algorithm, we choose the homogeneity method because its advantages are fast execution time (3 seconds for image of 512 pixels $\times$ 512 pixels) [9], low computational complexity and low error rate, and it detects the edge of image better than Sobel and Gaussian techniques especially in the noise

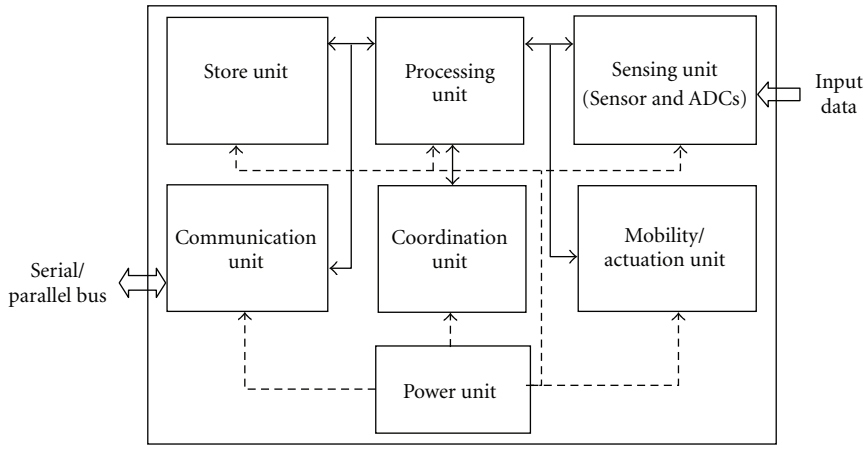

Figure 1: A common multimedia sensor node architecture.

condition [5–9]. Since the method relies on color signatures, it can be applied for many type images such as black and white, greyscale, or color images. We only need to specify the distance between two points of image. As a result, the resulting signatures would be perceptually meaningful. The details of the detection technique is explained in Section 3.1. Based on the algorithm, we propose two efficient energy schemes that are suitable for resource-constrained in WVSNs.

The remainder of this paper is organized as follows. In Section 2, we discuss related work and present general image and video compression for WVSNs. In Section 3, two proposed schemes based on video compression for WVSNs are introduced. In Section 4, we critically evaluate and analyze some simulation results. Finally, our conclusions and suggestions for future work are given in Section 5.

2. Related Work

2.1. Sensor Node Structure. A WSN is a group of tiny disposable wireless devices. Each device, referred to as a sensor node, is equipped with sensors/actuators, microprocessors, communication, and power modules. If the data transferred are multimedia, each sensor, called a multimedia sensor, should be equipped with the functions to capture, process, and transfer the data. We can categorize WVSN platforms into three categories based on the computation power, namely lightweight, intermediate, and PDA-class platforms [13] or based on purpose, namely general, heavily coupled, and externally dependent architectures [3]. Since the visual data require higher bandwidth usage due to the amount of data to be transmitted and higher power consumption due to the complexity of coding and vision processing algorithms, only PDA-class platform can be applied for real-time video applications that require the high data rate [1, 3, 13, 14]. The architectures of multimedia sensors are shown in Figure 1 [1, 3, 13, 14]. The multimedia sensor device includes seven basic components: a sensing unit, a processing unit (CPU), a communication unit, a coordination unit, a storage unit (memory), an optional mobility/actuation unit, and a power unit. There are two subunits in the sensing unit, sensors (cameras, microphones, and/or scalar sensors) and analog-to-digital converters (ADCs). The ADC subunit converts the analog signals, which are collected by the sensor subunit, into digital signals, and then transfers them into the processing unit. The processing unit carries out the system software to coordinate sensing and communication tasks, and interacts with the storage unit. The coordination unit performs the location management, motion controller, and network synchronization tasks. The communication unit manages the communication protocol stack, system software, and middleware. An optional mobility/actuation unit is used to move or manipulate the objects. Finally, the power unit supports energy for the whole system.

2.2. Image and Video Compression on WVSNs. The availability of inexpensive equipments such as CMOS cameras makes them possible to ubiquitously capture multimedia content from the environment and has fostered the development of image and video processing applications for WSNs [1, 4, 15–23]. Ahmad et al. [4] performed to evaluate energy efficiency of a predictive coding scheme for deployment on real-life sensor based on coding intraframes (I-frames). Their results show that the energy consumed for coding an I-frame (on average 60.03 mJ/frame) is much lower than that for coding an interframe (on average 763.68 mJ/frame), predictive frame (P-frame), or bidirectionally predicted frame (B-frame).

Magli et al. [20] proposed an algorithm for low-complexity video coding with video surveillance applications based on active regions. The results show that the quality of decoding video in the algorithm is competitive with that of MPEG-2 while its complexity is lower than that of MPEG-2. Chow [21] proposed a video-coding method based on motion compensation followed by shape compensation to replace the conventional discrete cosine transform (DCT) coding. However, the method is efficiently applied only for binary images. Yeo and Ramchandran [22] focused on exploiting the correlation among cameras and proposed two alternative models to capture interview correlation among

cameras with overlapping views. The results show that the quality of decoding intraframes in proposed models improves 2.5 dB compared with that of H.263+. Liu et al. [23] proposed a surveillance video compression system based on Wyner-Ziv coding that balances between computational complexity and coding efficiency. The results show that the proposed system not only increases the coding efficiency but also reduces the encoder complexity.

One of the papers most closely related to our work is [20]. In that paper, Magli et al. devised an algorithm that can quickly find active regions. Two types of scenes are considered in their algorithm, motion of background and foreground objects. The main point of the algorithm is to categorize the differences among noise, shadow, and illumination regions. Since the algorithm performs to compare all pixels in every 8×8 block, it takes a long time and consumes too much energy for processing per frame: the total encoding time is more than 1000 ms for one frame, and energy consumption for encoding one frame is 42 mJ in case the group of pictures (GOP) is 25 [20]. On the other hand, the quality of decoding video is not high since the method only allows the noise standard deviation with a maximum error of 0.25. In case of the motion with a foreground object, peak signal-to-noise ratio (PSNR) of this algorithm is less than 2 dB compared with that of MPEG-2 [20].

For the basic image coding techniques, we can divide the techniques into four categories, predictive coding, block transform coding, vector quantization, and subband and wavelet coding [24]. In these categories, we consider the second and fourth, that is, block transform coding, and subband and wavelet coding, due to their simple implementation. The DCT and discrete wavelet transform (DWT) techniques are two special image-compression techniques belonging to these categories. The advantage of DCT is that it can be easily implemented with relatively low memory requirements, whereas its disadvantage is its low compression and low bit rate, which results in annoying blocking and ringing artifacts. In contrast, the advantage of DWT is high compression rate and high image quality, whereas its disadvantages include its high computational complexity and substantial memory requirements [25–27].

To address the problems above, the authors [15–19] proposed the new methods to reduce the computational complexity and memory utilization. Chrysafis and Ortega [15] focused on reducing memory at both encoder and decoder in image compression by using a line-based approach for the implementation of the wavelet transform and storing only a local set of wavelet coefficients. As a result, the quality of decoding image and memory utilization are improved clearly. Lee et al. [16] optimized the JPEG implementations and measured the energy consumption for compressing and transferring image data. The results showed that the proposed algorithm can improve both the quality of image and energy consumption. Oliver and Malumbres [17] proposed an image compression algorithm to improve the efficient construction of wavelet coefficient trees. As a result, the proposed algorithm reduces not only memory requirement but also processing time. Oliver and Malumbres [18] proposed an algorithm to efficiently compute the two-dimensional wavelet transform in image compression. The results showed that the algorithm can reduce the memory requirement up to 200 times. Rein and Reisslein [19] presented an overview of the techniques using wavelet transform that allow for transforming images on in-network sensors with low memory. As analyzing above, we believe that two techniques (DCT and DWT) will be improved and applied for the multimedia applications on WSNs.

For the video coding paradigms, we can divide the techniques into two types, distributed source coding and individual source coding [28]. Both can be applied to three compression techniques, single-layer coding (i.e., JPEG), multilayer coding (i.e., JPEG2000), and multidescription coding. In this paper, we focus attention on the video coding paradigms. The details of three techniques can be seen in [28]. The three techniques have been inspected in the former type, but they have not yet been fully studied in the latter type [29–31].

For former type such as MPEG-X and H.26L, all compression processes are carried out at the source nodes. The nodes must thus implement all tasks of the video compression process, including transforming, finding motion vectors and compensating for motion, and encoding. Then they send compressed data using a multihop technique or directly to the base station. The advantage of this method is that since the source nodes do not need to communicate with the other nodes in the encoding process, the method is simple to perform. Otherwise, the disadvantage of the method is that the source nodes will be exhausted quickly because of overloading. Therefore, the energy distribution in the network is not balanced, and the network lifetime will be reduced.

To solve this problem, researchers concentrate attention on latter type. Wyner-Ziv encoder [29] and Power-efficient Robust hIgh-compression Syndrome-based Multimedia coding (PRISM) are two typical schemes for distributing source coding on WVSNs [22, 32]. In these schemes, researchers either divide video data into small blocks that are suitable to be performed by several nodes or try to reduce the encoding complexity at the source nodes. As a result, the energy consumption of network is balanced, and the network lifetime is prolonged. However, the schemes have some disadvantages. Since the nodes need to communicate with one another, they must spend a part of their energy on communicating. In some cases, the schemes will not achieve high compression efficiency if there are only few complex encoders [33]. Otherwise, the quality of decompressed data is reduced due to wireless channel errors.

To address this problem, researchers use either the channel codes that are able to protest against channel errors, such as low-density parity check code (LDPC) [30] and Turbo code [29], or a backward channel to improve the quality of reference frames [23]. In the paper, we consider combining individual source coding and distributed source coding for the proposed method. In our algorithm, we perform to compare the edges of objects among the frames to find the active regions quickly. We try to reduce the complexity of the encoder by sharing the processing tasks

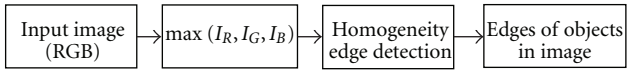

Figure 2: The homogeneity edge detection method.

such as detecting edges of objects, finding motion vectors, and compensating for motion to other nodes. To the best of our knowledge, using the edges of objects in the frames to compress video on WVSNs has not been considered in the literature.

3. Proposed Schemes

3.1. Homogeneity Edge Detection Technique. In the proposed algorithm, we use the homogeneity edge detection, based on the color edge detection algorithms, to estimate motion (find motion vectors) and compensate for motion. We can divide the color edge detection algorithms into three categories, namely output fusion method, multidimensional gradient method, and vector method [6, 8].

In the output fusion method, finding edge detection is performed independently for three color components, red, green, and blue (RGB), and then three edges are combined into the final edge. In the multidimensional gradient method, three color components are first combined before performing edge detection. In the vector method, the edges of images are found based on the vector nature of colors and their rotation. (See [6] for more details on the three techniques.)

Among three methods, we consider the output fusion method and the multidimensional gradient method due to their simple implementation. The advantage of output fusion method is simple to perform, whereas its disadvantage is to consume more energy for detecting edges because it has to perform edge detection three times [6, 8]. The edge detection task includes many steps and consumes a lot of energy [5, 7], and thus output fusion method is not suitable to implement on wireless sensor devices. To solve the problem, we first perform preprocessing three color components before implementing edge detection, and then apply the homogeneity edge detection method in [7, 9]. As a result, we only carry out edge detection one time, and thus save energy consumption of wireless sensor devices.

Figure 2 shows the details of steps of homogeneity edge detection method. In Figure 2, we modify the conventional homogeneity edge detection method by inserting the preprocessing block before detecting edges of objects in images to reduce the complexity computation. Therefore, the detection process includes two steps as follows.

Step 1 (preprocessing). In this step, we select one of three color elements of input image to detect as follows

$$I(i,j) = \max(I_R(i,j), I_G(i,j), I_B(i,j)), \quad (1)$$

where $I_R(i,j), I_G(i,j), I_B(i,j)$ are the intensity of red, green, and blue elements of input image at pixel (i,j), respectively. The goal of the step is to reduce energy consumption of wireless sensor devices for detecting edges of objects by carrying out edge-detection task one time.

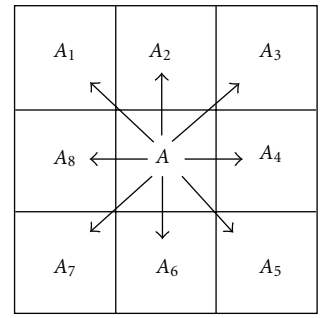

Figure 3: The implementation of homogeneity operator.

Step 2 (detecting edges of objects). We use the homogeneity operator that performs to calculate the different value of center point with eight neighbors to find edges of objects in image (E_A), as shown in Figure 3. The homogeneity operator based on [9] is defined as

$$E_A(i,j) = \begin{cases} \max\{\,|I_A(i,j) - I_{A_n}(i,j)\,| \, n = 1,2,\ldots,8\}, \\ \text{if } \max\{\,|I_A(i,j) - I_{A_n}(i,j)\,|\,\} \geq \text{threshold}_1, \\ 0, \quad \text{otherwise} \end{cases}$$

$$0 \leq \text{threshold}_1 \leq 255, \quad (2)$$

where threshold_1 is the threshold that is used to improve the quality of edges of objects in image. There are several ways to determine the threshold. Observing the edges of objects for a set of tested images and selecting the value that gains acceptable edges of objects is the simple way [34]. In our simulation, we choose $\text{threshold}_1 = 90$.

3.2. Proposed Video Compression Algorithm. We make some network assumptions based on [1, 3, 20] in our proposed algorithm as follows. (1) The transmission range of the sensor nodes can be adjusted dynamically to allow multihop communication within a cluster in WVSNs. (2) Source nodes can load the raw video data into their memory. (3) All nodes are able to perform the complex tasks. (4) The energy consumption for *SYN/ACK* packets is not considered. (5) Since sensors are limited by their storage and processing capacity, we assume that video input includes only I- and P-frames. (6) We consider scenes with small changes in the backgrounds and low motion of objects.

Our proposal is based on the algorithm in [20]. As we analyzed in Section 2, the most difficult problem of the algorithm in [20] is that it has to compare all pixels in every 8×8 block to determine active regions, and thus, it takes a long time to scan and consumes too much energy per frame. To solve this problem, we propose a method to find motion regions based on comparing the edges of objects among frames.

The proposed algorithm has three different points from [20]. First, we use the difference of edges of objects among frames to mark motion regions while [20] uses the differences among noise, shadow, and illumination regions. Since we use edges of objects, processing time and energy consumption for encoding frames of our method are less than those of [20]. Secondly, we use edges of objects in the background images to increase accuracy when performing to mark motion regions. As a result, the numbers of motion vectors and motion regions reduce, and thus compression rate is improved. Thirdly, we then apply our method (finding the motion regions by comparing the edges of objects) for MPEG-2/H.262 encoder that is suitable to perform for wireless applications because of its low complexity of algorithm and acceptable quality of decoding data [35]. The main difference between the proposed algorithm and MPEG-2 is that only the edges of objects in the frames are required in the proposed method while MPEG-2 requires all data of the frames. Therefore, we can save energy and time for finding motion vectors and compensating for motion.

Figure 4 depicts an overview of the proposed video compression system. First, the source node captures the current frame to detect its edges of objects and compares the detected edges of objects with the edges of objects in the background images. The goal of this step is to reduce noise. Then the detected edges of objects of this frame are stored in the buffer of source node. The source node repeats the same process for the next frame and compares the detected edges of objects of this frame with those of the previous frame in its buffer to mark active regions. Based on the active regions, the source node finds motion vectors and compensates for motion. Finally, the motion regions are transformed, quantized, run length encoding (RLE), and Huffman encoding, respectively, and the motion vectors are encoded by RLE, and Huffman encoding by the encode block.

Figure 5 depicts the details of the proposed video compression algorithm using the homogeneity edge detection technique. First, video data are captured by the source node. The video data consist of a sequence of frames, which are encoded as I-frames and P-frames, where I-frame is the frame which stores only information within the current frame and P-frame is the frame which stores the difference (motion vectors and motion regions) among one or more neighboring frames. These frames are stored in the frame buffer block. At the block, an input frame is checked whether it is I-frame or P-frame. If the input is an I-frame, it is compressed by the following process: DCT, quantization, RLE and Huffman encoding. On the other hand, if the input is a P-frame, it will be transferred to the edge detector block to mark active regions, as shown in Figure 6. In the figure, we perform two steps to determine the active regions as follows.

Step 1 (comparing edges of objects of frames). First, the edge-detector block performs to detect edges of objects in the frames (P_1, P_2 frames, and background images). At the initial time, we assume that the background images were stored in the buffer. At the next times, we will use the previous frames as the background images. The detected edges of objects among frames are then compared each other. The different edges of objects between two frames are calculated as

$$\Delta E_{P_{12}}(i,j) = \begin{cases} 0, & \text{if } E_{P_1}(i,j) = E_{P_2}(i,j), \\ E_{P_2}(i,j), & \text{otherwise.} \end{cases} \tag{3}$$

Step 2 (marking the active regions). In this step, the number of pixels ($N_{\Delta P_{12}}$) whose value is different from zero in each block of ΔP_{12} frame is calculated. If $N_{\Delta P_{12}}$ is more than threshold$_2$, which depends on the size of block, the block is marked. In our simulation, threshold$_2$ is set up as 32 if the size of block is 8×8.

The algorithm then performs to find motion vectors at the motion estimation block and compensate for motion at the motion compensation block in Figure 5 based on the marked regions (active regions). To save energy consumption for finding motion vectors, we use a "three-step search" algorithm. The algorithm searches motion vector based on comparing and determining minimum mean absolute errors (MAEs) of eight points that have the same distance from center point (estimated point). An example of "three-step search" is shown in Figure 7. In the figure, the motion vector *AD* that has minimum MAE is determined after performing to search three steps. The details of steps of the algorithm can be seen in [36, 37]. Finally, the motion regions are transformed DCT, quantized, RLE, and Huffman encoding, respectively, and motion vectors are encoded by RLE and Huffman encoding. Based on the proposed video compression algorithm, we propose two different schemes to implement this video compression algorithm in WVSNs.

3.3. Energy Evaluation Model. For evaluating energy consumption, we use the wireless communication energy model proposed in [38–40]. The energy consumption in transmission per bit is

$$e_{tx} = \begin{cases} e_{\text{elec}} + \varepsilon_{fs} d^2, & d < d_0, \\ e_{\text{elec}} + \varepsilon_{mp} d^4, & d \geq d_0, \end{cases} \tag{4}$$

and the energy consumed in reception per bit is

$$e_{rx} = e_{\text{elec}}, \tag{5}$$

where d_0 is the close-in referential distance or threshold distance, which is determined from measurements close to the transmitter, and d is the distance between the wireless transmitter and the receiver. The notation e_{elec} is the energy consumption by the circuit per bit, and $\varepsilon_{fs} d^2$ or $\varepsilon_{mp} d^4$ is the energy amplifier that depends on the transmitter amplifier model.

In the proposed video compression algorithm, we process data following 8×8 blocks. Therefore, we model the energy consumption for blocks. For I-frame, the energy consumption in video compression for the ith block of I-frame (E^i_{Icp}) is

$$E^i_{Icp} = E^i_{\text{DCT}_I} + E^i_{\text{Code}_I} = a_i e_{\text{DCT}} + b_i e_{\text{Code}}, \tag{6}$$

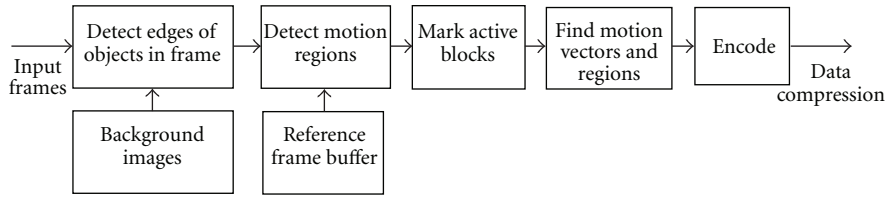

Figure 4: The proposed video compression algorithm.

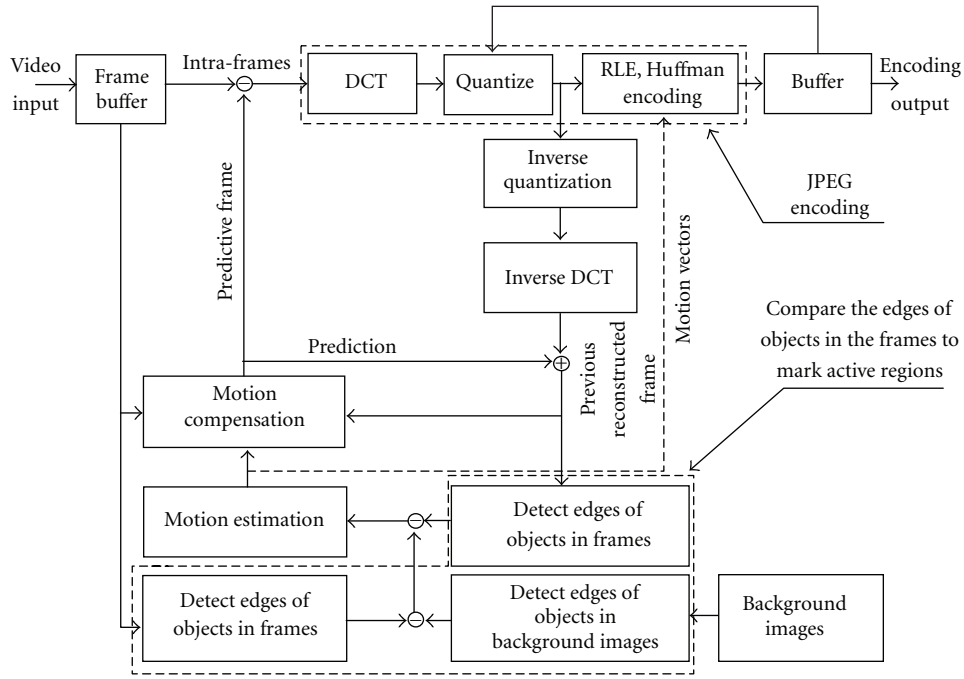

Figure 5: The details of proposed video compression algorithm based on detecting the edges of objects in the frames.

where $E^i_{\mathrm{DCT}_I}$ and $E^i_{\mathrm{Code}_I}$ are the energy dissipation for transforming DCT and coding for the ith block of I-frame, respectively. a_i and b_i are the data sizes of blocks before transforming DCT and coding (quantizing and coding) for the ith block of I-frame, respectively. e_{DCT} and e_{Code} are the energy consumption per bit for transforming DCT and coding, respectively. Consequently, the energy consumption for compressing I-frame (E_{Icp}) is

$$E_{Icp} = \sum_{i=1}^{N} E^i_{Icp}, \tag{7}$$

where N is the number of blocks of frame. Total energy consumption for I-frame (E_I) is

$$E_I = E_{Irx} + E_{Icp} + E_{Itx} = Ae_{rx} + E_{Icp} + Ce_{tx}, \tag{8}$$

where E_{Irx} and E_{Itx} are the energy consumed for receiving and transmitting per I-frame, respectively. A is the data size of receiving I-frame, and C is the data size of transmitting I-frame after coding.

For P-frame, the energy consumption for compressing the ith block of P-frame (E^i_{Pcp}) is

$$E^i_{Pcp} = E^i_{\mathrm{DCT}_P} + E^i_{\mathrm{Code}_P} = a'_i e_{\mathrm{DCT}} + b'_i e_{\mathrm{Code}}, \tag{9}$$

where a'_i and b'_i are the data sizes of blocks before transforming DCT and coding (quantizing and coding) for the ith block of P-frame, respectively. Therefore, the energy consumption for compressing P-frame (E_{Pcp}) is

$$E_{Pcp} = E_{\mathrm{Detect}} + E_{\mathrm{Mot}} + \sum_{i=1}^{N} E^i_{Pcp}, \tag{10}$$

where E_{Detect} is the energy dissipation for detecting the edges of objects in previous and current frames, and E_{Mot} is the energy dissipation for finding motion vectors and motion regions. Total energy consumed for P-frame (E_P) is

$$E_P = E_{Prx} + E_{Pcp} + E_{Ptx} = A' e_{rx} + E_{Pcp} + C' e_{tx}, \tag{11}$$

where E_{Prx} and E_{Ptx} are the energy consumed for receiving and transmitting per P-frame, respectively. A' is the data size of receiving P-frame, and C' is the data size of transmitting P-frame after coding.

The value of E_{Detect} and E_{Mot} are much larger than e_{DCT} and e_{Code} in our proposed algorithm because it has to scan the entire frame to find its edges of objects and to scan active regions to determine motion vectors and motion regions. The details of steps for calculating the energy values are

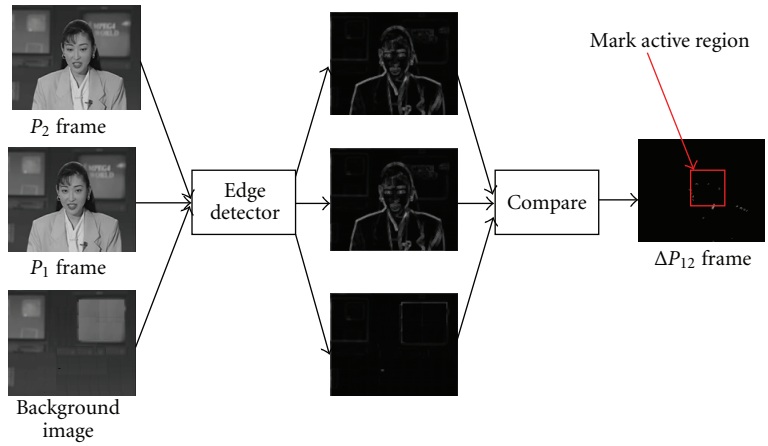

Figure 6: Comparing the edges of objects in the frames to mark active regions.

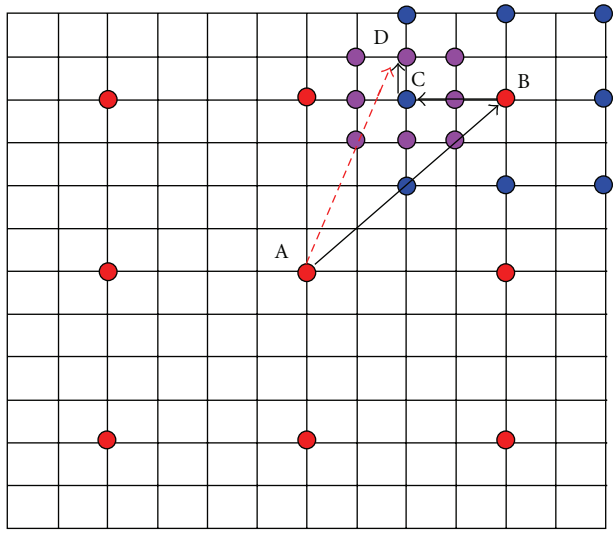

Figure 7: An example of "three-step search" algorithm.

explained in Appendix A. In the proposed algorithm, we therefore distribute these two tasks to other nodes instead of performing to compress video data at the source nodes.

3.4. Proposed Video Compression Scheme 1. Figure 8 depicts the first proposed scheme with a normal case of the video compression system in a WVSN, where the number of hops from the source node to the cluster head is three. Figure 8(a) depicts a case where the input frame is an I-frame. For the frame, we compress by the following process: DCT, quantization, and encoding along the path from the source node to the cluster head. Total energy consumption for I-frame is

$$\begin{aligned} E_I(\text{scheme 1}) &= E_{Irx} + E_{Icp} + E_{Itx} \\ &= E_{Irx} + \sum_{i=1}^{N} \left(E_{\text{DCT}_I}^{Ti} + E_{\text{Code}_I}^{Ci} \right) + E_{Itx}, \end{aligned} \tag{12}$$

where $E_{\text{DCT}_I}^{Ti}$ is the energy dissipation for transforming DCT at transforming node T for the ith block of I-frame, and $E_{\text{Code}_I}^{Ci}$ is the energy consumption for coding at coding node C for the ith block of I-frame.

Figure 8(b) depicts the other case, where the input frame is an P-frame. Node S must first detect the edges of objects in the frames and those of the background images, and then compare them to each other to mark active regions. The step is performed similar to the Steps 1 and 2 in Section 3.2. Based on the active regions, node S finds motion vectors and compensates for motion. Then the motion regions are transformed DCT, quantized, RLE and Huffman encoding, respectively, and the motion vectors are encoded by RLE and Huffman encoding. In this case, node T performs the DCT task, and node C performs the coding task. By sharing the compression tasks among several nodes in a cluster, the energy consumption of the nodes will be balanced. Total energy consumption for P-frame is

$$\begin{aligned} E_P(\text{scheme 1}) &= E_{Prx} + E_{Pcp} + E_{Ptx} \\ &= E_{Prx} + E_{\text{Detect}_{12}}^{S} + E_{\text{Mot}}^{S} \\ &\quad + \sum_{i=1}^{N} \left(E_{\text{DCT}_P}^{Ti} + E_{\text{Code}_P}^{Ci} \right) + E_{Ptx}, \end{aligned} \tag{13}$$

where $E_{\text{Detect}_{12}}^{S}$ is the energy dissipation for detecting the edges of objects in previous and current frames at node S, E_{Mot}^{S} is the energy dissipation for finding motion vectors and motion regions, $E_{\text{DCT}_P}^{Ti}$ is the energy dissipation for transforming DCT at node T for the ith block of P-frame, and $E_{\text{Code}_P}^{Ci}$ is the energy spent for coding at node C for the ith block of P-frame.

When the number of hops between node S and node C, say h_{sc}, is less than three, the tasks cannot be fully distributed as shown in Figure 8. Accordingly, all the compression tasks (DCT and coding) are allocated to the intermediate node when $h_{sc} = 2$; and all are done on the source node when $h_{sc} = 1$.

We recognize that in scheme 1, source nodes have to implement many tasks (to detect the edges of objects in

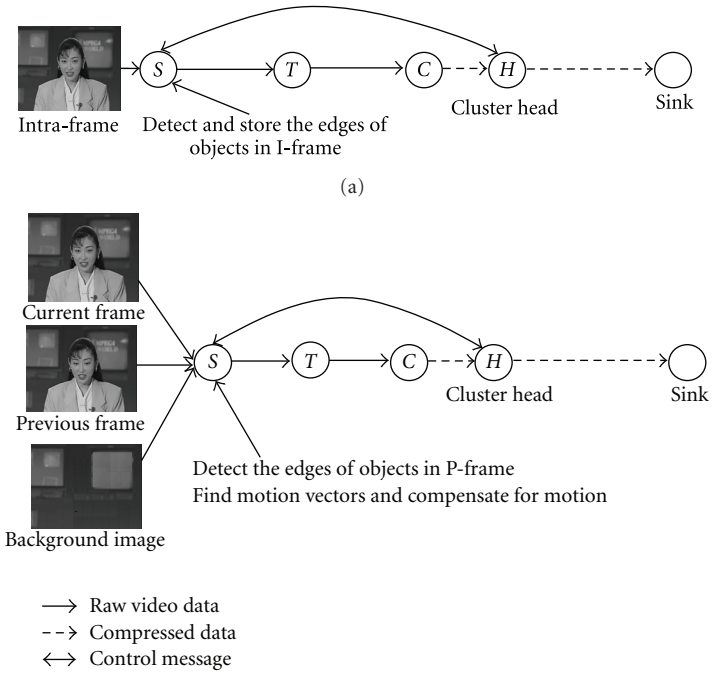

Figure 8: The normal case of the proposed video compression scheme 1 with $N_{\text{hops}} = 3$ for (a) I-frames and (b) P-frames.

the frames, to compare them, to find the motion vectors and compensate for motion, to detect the edges of objects in the background images, and compare and store edges of objects), while other nodes still have enough energy to do these tasks. Therefore, we need to distribute the tasks, for example, detecting edges of objects in the frames and finding motion vectors and motion regions, to reduce overload at the source nodes. To solve the problem, we propose the second scheme.

3.5. Proposed Video Compression Scheme 2. Our second proposed scheme is shown in Figure 9. In this scheme, we improve scheme 1 and use the encoding technique based on the Wyner-Ziv encoder to reduce the complexity of encoder [4, 23, 41]. In Figure 9, the sequence of frames of the input video is divided into two groups, P-frames and I-frames. The different points of the proposed scheme 2 from the proposed scheme 1 are to detect edges of objects in I-frame, to find motion vectors and compensate for motion at the next to source node *S*. Therefore, we reduce overload at the source node. The details of steps of proposed scheme 2 are performed as follows.

For I-frames, we use the conventional encoder (i.e., H.262, H.263, H.263+, or H.264 encoders). In our simulation, intraframes are encoded by H.262 encoder that is suitable for wireless application because of its low complexity of algorithm and acceptable quality of decoding data [35]. The frames will be used as reference frames to find the motion vectors and compensate for motion.

For P-frames, we implement five steps to estimate and compensate for motion, as shown in Figure 9.

Step 1. The current frame is detected to find its edges of objects at the encoder (node *S*). Then it is compared with the edges of objects in the background images to cut down noise. The difference data between the edges are transformed, quantized, and encoded before being sent to the decoder (node *T*).

Step 2. At the decoder, we perform decoding and inversing transformation to rebuild the difference data between the edges. The difference data are compared with the edges of objects in the reference frame at the decoder to mark active regions. The step is performed similar to the Steps 1 and 2 in Section 3.2.

Step 3. The indexes of marked active regions are sent back to the encoder.

Step 4. The encoder sends only active regions based on the indexes of marked active regions to the decoder.

Step 5. The decoder estimates motion (motion vectors) and compensates for motion based on active regions and reference frame. Finally, the motion regions will be transformed, quantized, and encoded by JPEG-encoding block. By reducing the computational complexity of the encoder, the video compression tasks are shared by both the encoder and the decoder.

Figure 10 depicts an example of the proposed scheme 2 where the number of hops from the source node to the cluster head is three. Figure 10(a) illustrates a situation where the input frame is an I-frame. For the frame, the data are

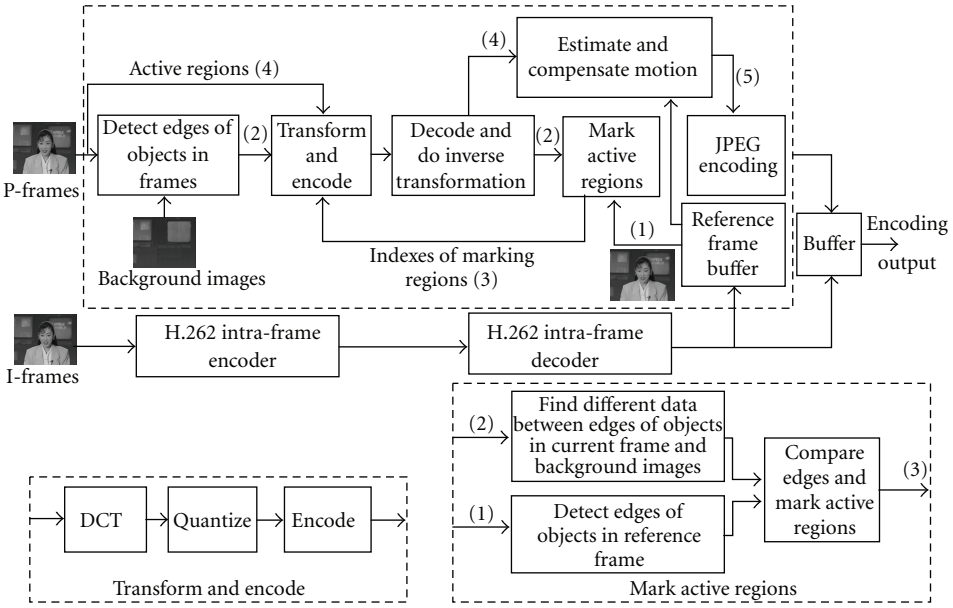

Figure 9: The proposed video compression scheme 2.

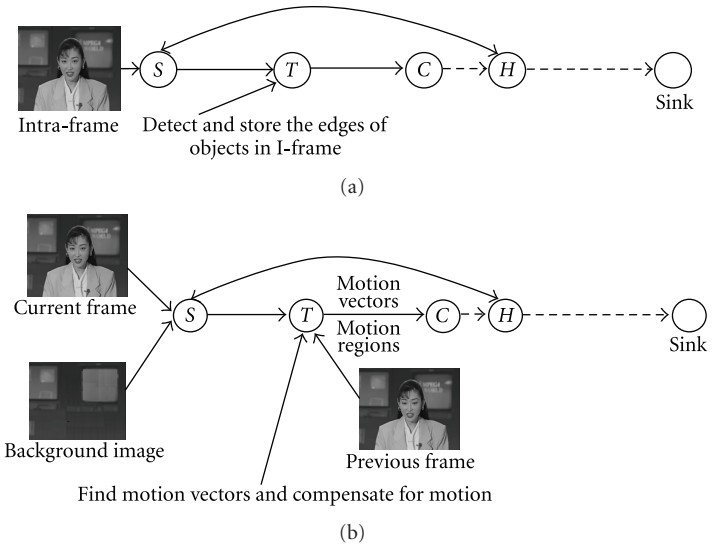

Figure 10: The normal case of the proposed video compression scheme 2 with $N_{\text{hops}} = 3$ for (a) I-frames and (b) P-frames.

compressed by H.262 encoder along the path from the source node to the cluster head. In the figure, node T not only performs the transforming task but also detects and stores the edges of objects in the I-frames for motion estimation and motion compensation. The energy consumption in video compression for I-frame is

$$\begin{aligned} E_I(\text{scheme 2}) &= E_{Irx} + E_{Icp} + E_{Itx} \\ &= E_{Irx} + E^T_{\text{Detect}} \\ &\quad + \sum_{i=1}^{N} \left(E^{Ti}_{\text{DCT}_I} + E^{Ci}_{\text{Code}_I} \right) + E_{Itx}, \end{aligned} \tag{14}$$

where E^T_{Detect} is the energy dissipation for detecting the edge of I-frame at node T.

Figure 10(b) shows the compression scheme for a P-frame. Node S first performs edge detection on the current frame. Then the edges of objects are compared with the edges of objects in the background images. After the comparison, the difference data will be sent to the node T. Node T compares the difference data with the edges of objects in the reference frame to mark active regions and sends indexes of active regions back to node S. Based on the indexes, node S sends the active regions of current frame (P-frame) to node T to estimate and compensate for motion (motion vectors and motion regions). Then the motion regions will be transformed, quantized, RLE and Huffman encoded,

respectively, and the motion vectors are encoded by RLE and Huffman encoding. In this case, node T performs the DCT task, and node C performs the coding task. Therefore, the energy consumption in video compression for P-frame is

$$\begin{aligned} E_P(\text{scheme 2}) &= E_{Prx} + E_{Pcp} + E_{Ptx} \\ &= E_{Prx} + E^S_{\text{Detect}} + E^T_{\text{Mot}} \\ &\quad + \sum_{i=1}^{N} \left(E^{Ti}_{\text{DCT}_P} + E^{Ci}_{\text{Code}_P} \right) + E_{Ptx} + E^S_{Dtx} + E^S_{Mtx}, \\ E^S_{Dtx} &= De_{tx}, \\ E^S_{Mtx} &= D' e_{tx}, \end{aligned} \tag{15}$$

where E^S_{Detect} is the energy dissipation for detecting the edge of P-frame at node S, E^S_{Dtx} is the energy dissipation for transferring the edges of objects in current frame from node S to node T, and E^S_{Mtx} is the energy dissipation for transferring the marking regions of current frame from node S to node T. D and D' are the data sizes of the edges of objects in current frame and the marking regions of current frame, respectively. Since the edges of objects in frames have the high correlation, we compress the data before sending to the node T.

In scheme 2, since we use the encoding technique that uses active regions, only a part of current frame based on marking regions is transferred to estimate and compensate for motion as shown in Figure 9. Therefore, the energy (E^S_{Mtx}) is much smaller than the energy consumption for sending the whole frame. When the number of hops between node S to node C, h_{sc}, is less than three, the tasks will be allocated to the same node like the scheme 1, due to the lack of intermediate nodes.

4. Simulation Results

4.1. Simulation Setup. We used Visual Studio C to design our simulation. In our simulation, we consider six-sensor networks with sizes of 100, 200, 300, 500, 800, and 1000 nodes, randomly distributed over a 500 m × 500 m field. The source nodes are randomly selected. We divide the wireless network into many parts (clusters), and each part is controlled with a special node termed a cluster head node. We choose the cluster head based on LEACH-C [38]. We choose the energy model parameter values in (4) and (5) as follows: $e_{\text{elec}} = 50\,\text{nJ/bit}$, $\varepsilon_{fs} = 0.01\,\text{nJ/bit/m}^2$, $\varepsilon_{mp} = 0.0000013\,\text{nJ/bit/m}^4$, and $d_0 = 100\,\text{m}$, using the typical values in the previous literature [38–40]. Based on [39, 40], we select the values of the parameters for computing energy model as follows: $e_{\text{DCT}} = 20\,\text{nJ/bit}$ and $e_{\text{Code}} = 90\,\text{nJ/bit}$. Based on [5, 7, 36, 40, 42], we calculate the energy for homogeneity edge detection $E_{\text{Detect}} \approx 266846\,\text{nJ/frame}$ and the energy for finding motion region $e_{\text{Mot}} \approx 1205\,\text{nJ/bit}$.

We select the node that is closest to the center of the field as the base station. Every sensor is provided with two joules as startup energy. We use the *Akiyo* video whose background images change slowly, which is used in relevant video compression literature, supported by the quarter common interchange format (QCIF, 176 pixels × 144 pixels), with 24 bits/pixel and 150 frames as input data. We use MPEG-2/H.262, the main profile at low level (SP@LL) that is suitable for wireless applications [35]. In our simulation, the number of reference background images is five, and GOP is five (IPPPPIPPPP...). The reference background images are updated and stored in the buffer of the video sensors. These parameters are suitable for the buffer of video sensors in a wireless network [28].

4.2. Simulation Results. To evaluate the effect of different parameters on the execution of the proposed video compression schemes, several simulations are conducted. Two proposed schemes, which compress video data along the path from a source node to a cluster head node and transfer to the base station with a multihop transmission, are compared with MPEG-2/H.262 and the algorithm that compresses video data at the source node and transfers to the base station directly in [20].

We evaluate three parameters, the quality of decoding video (compression rate, video quality, and encoding time), the quality of the network (the numbers of received and lost frames), and the power consumption of the network in two cases. In the first case, we assume that there is no error on the channel. In the second case, we assume that there is an error-control scheme based on [43, 44]. In this case, we assume that there is only error between two nodes and do not consider interference with other nodes [43]. Therefore, the packet loss rate depends on the distance between two nodes. We assume that number of bits/packet for transmission is 8 [44]. A frame will be discarded if one or more packets constituting the frame are lost. Therefore, the average frame loss rate is defined as follows:

$$\text{average frame loss} = \frac{\text{number of frame loss}}{\text{number of received frames}}. \tag{16}$$

To evaluate the video compression algorithms, three parameters, PSNR, compression rate, and encoding time of decoding video, are used. Image quality is measured by PSNR metric as the following equation:

$$\text{PSNR [dB]} = 20 \log_{10} \frac{2^b - 1}{E||x_1(i,j) - x_2(i,j)||}, \tag{17}$$

where $x_1(i,j)$ is the value of pixel (i,j) in the original image, $x_2(i,j)$ is the value of pixel in the decompressed image, and b is the number of bits per pixel for the original image. The storage capacity is measured by the compression rate. It is defined as follows:

$$\text{compression rate} = \frac{\text{size of encoded image}}{\text{size of original image}}. \tag{18}$$

The PSNR and the compression rate have relation with each other. If the compression rate metric decreases, the image quality will go down. It means that PSNR will decrease. Thus, we need to balance two parameters

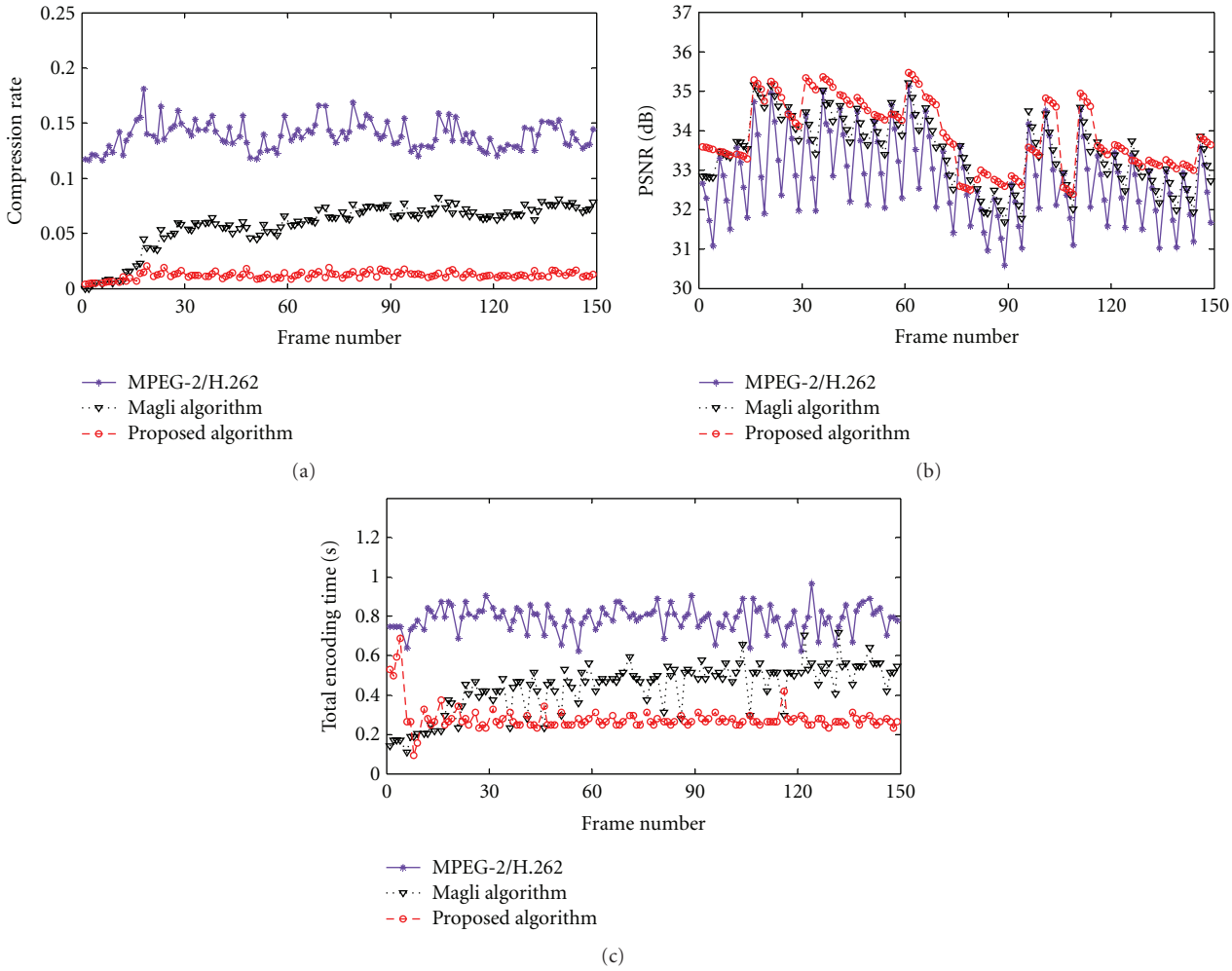

Figure 11: Simulation results with the *Akiyo* video: (a) the compression rate, (b) the quality of video, and (c) the total encoding time.

when compressing image, while researchers focus on the compression rate metric because of the energy constraints.

Comparison of decoding video quality: We perform to compress and decompress video data at a source node and calculate three parameters, PSNR, compression rate, and encoding time of decoding video, with two types of input videos, *Akiyo* and *Carphone* videos. The results are shown in Figures 11 and 12.

In Figure 11, when the background images of *Akiyo* video change slowly, the average quality of decoding video by the proposed algorithm is improved up 2 dB and 1 dB compared with the results by MPEG-2/H.262 and the Magli algorithm in [20], respectively, as shown in Figure 11(b) while the compression rate of decoding video by the proposed algorithm is smaller than that by other algorithms, as shown in Figure 11(a). On the other hand, the encoding time of the proposed algorithm is competitive with the Magli algorithm in [20] and lower than that of MPEG-2/H.262, as shown in Figure 11(c).

In Figure 12, when the background images of *Carphone* video change quickly, the average quality of decoding video by the proposed algorithm is not improved. In this case, the quality of decoding video by the proposed algorithm is not as good as that by MPEG-2/H.262 and the Magli algorithm in [20] for some frames, as shown in Figure 12(b) while the encoding time of the proposed algorithm is longer than that of MPEG-2/H.262, as shown in Figure 12(c).

Comparison of network quality: To evaluate the network quality, we perform to compress and transfer data video from source nodes to the base station. First, a video node, which is a source node, has to find the shortest way to the cluster head to send video data. Then the cluster head will send the compressed data to the base station using the multihop technique through other cluster heads. The simulation will stop when all source nodes depleted their energy.

First, we evaluate the energy consumption for encoding frame at the source node among MPEG-2/H.262, Magli algorithm in [20], and the proposed algorithm. The simulation results are shown in Figures 13 and 14. Although the energy consumption for encoding intraframes in the proposed algorithm is higher than that of other algorithms because of performing edge detection in the proposed algorithm as shown in Figures 13(a) and 14(a), the energy consumption for encoding interframes in the proposed algorithm is lower

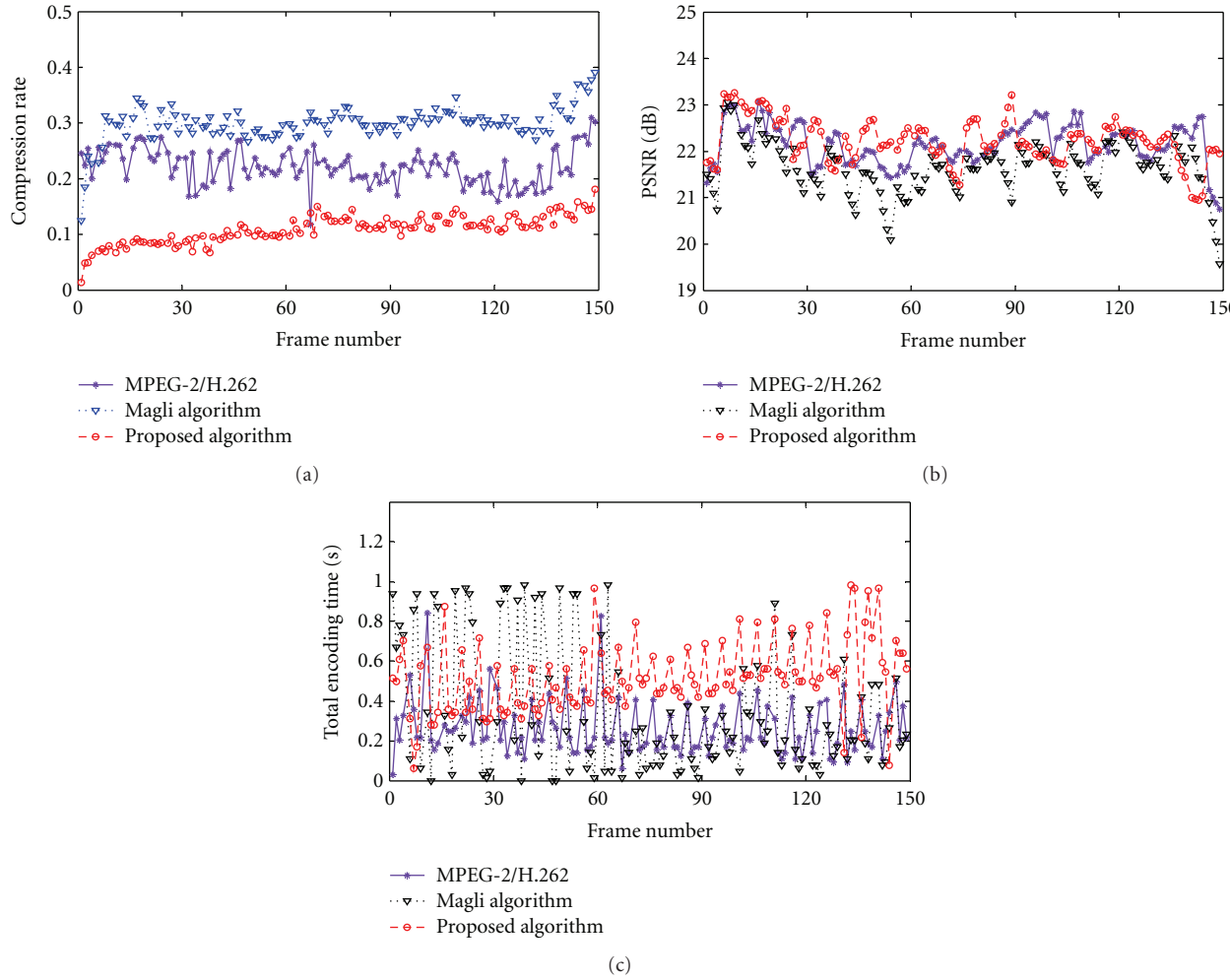

Figure 12: Simulation results with the *Carphone* video: (a) the compression rate, (b) the quality of video, and (c) the total encoding time.

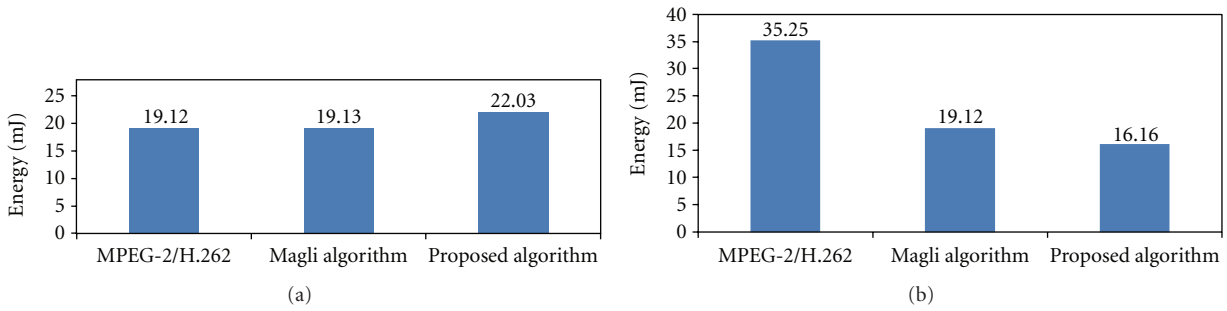

Figure 13: Comparing encoding energy consumption among algorithms for *Akiyo* video with: (a) the intraframe and (b) the interframe (P-frame).

than that of other algorithms due to reducing the complexity of motion estimation and motion compensation in the proposed algorithm, as shown in Figures 13(b) and 14(b). Therefore, total energy consumption for encoding intra and interframes in proposed algorithm is much lower than that of other algorithms because the number of interframes is much larger than the number of intraframes in video data, especially when GOP increases.

Then, we evaluate the network quality in two cases. In the first case, we assume that there is no error on the channel. In the second case, we assume that there is an error-control scheme. The simulation results are shown in Figures 15

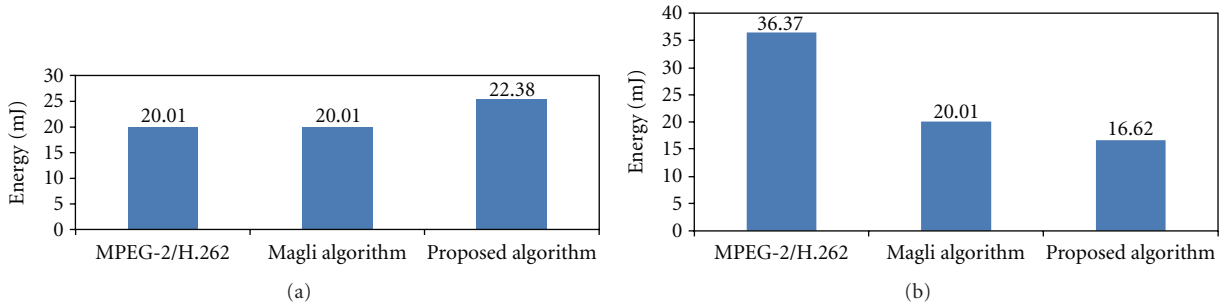

FIGURE 14: Comparing encoding energy consumption among algorithms for *Carphone* video with: (a) the intraframe and (b) the interframe (P-frame).

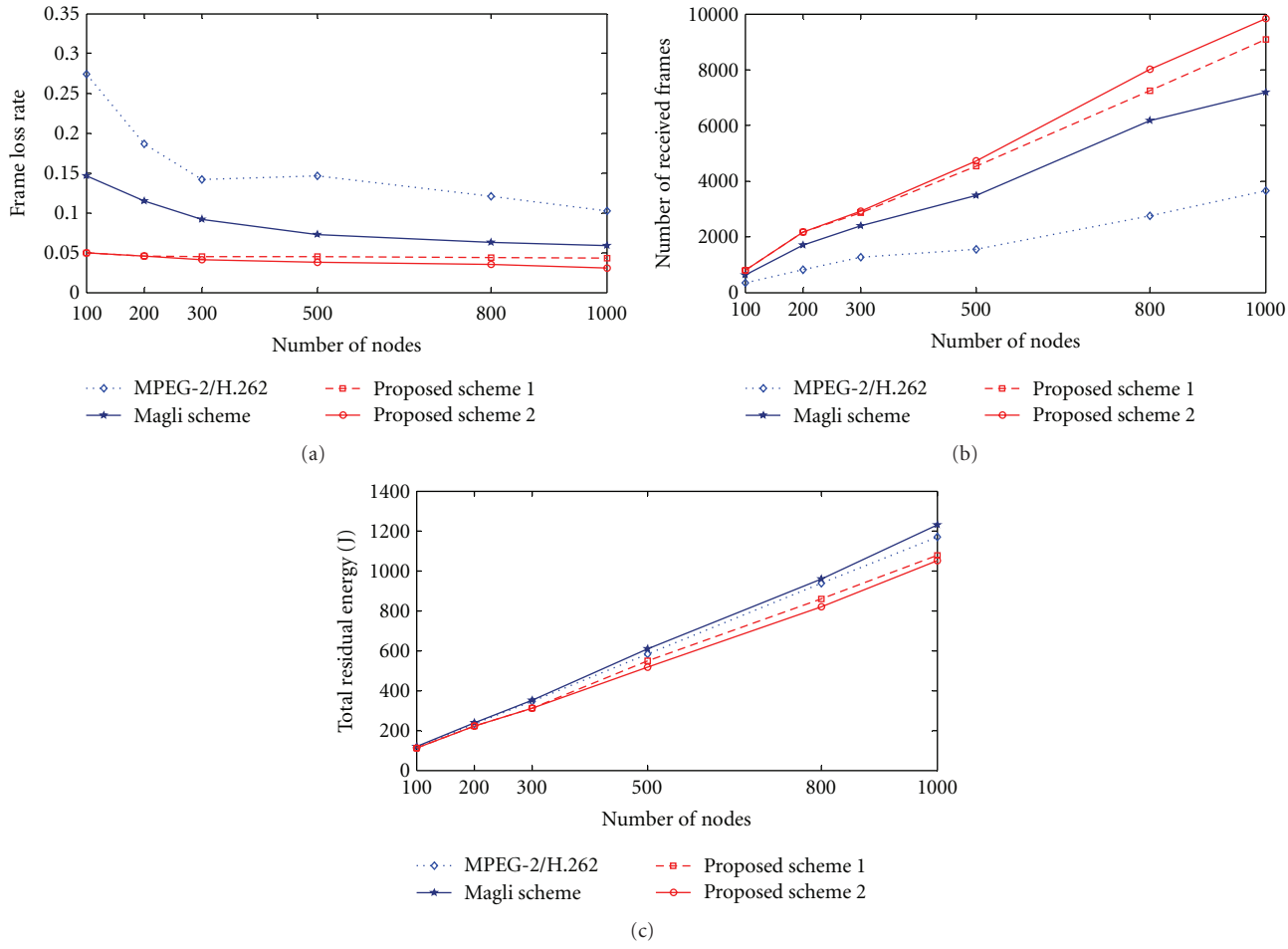

FIGURE 15: The results obtained for four schemes with QCIF *Akiyo* video in case of not having channel error: (a) the average frame loss rate, (b) the number of received frames, and (c) the total residual energy.

and 16. In both cases, when the number of nodes becomes large enough, the topology of the network will be improved. Thus, the frame loss rates are less than 5 percent with 1000 nodes as shown in Figure 15(a), and less than 10 percent with 1000 nodes as shown in Figure 16(a) in two proposed schemes. As a result, as shown in Figures 15(b) and 16(b), the numbers of received frames by the proposed schemes that perform to distribute the video compression tasks for multiple nodes from a source node to a cluster head are greater than those by the other schemes that perform to centralize the video compression tasks at a source node, while the energy consumptions of all schemes are almost the same, as shown in Figures 15(c) and 16(c).

Comparison of power consumption: To evaluate the energy balance, we also perform in two cases, not having error control scheme and having error control scheme. We measured

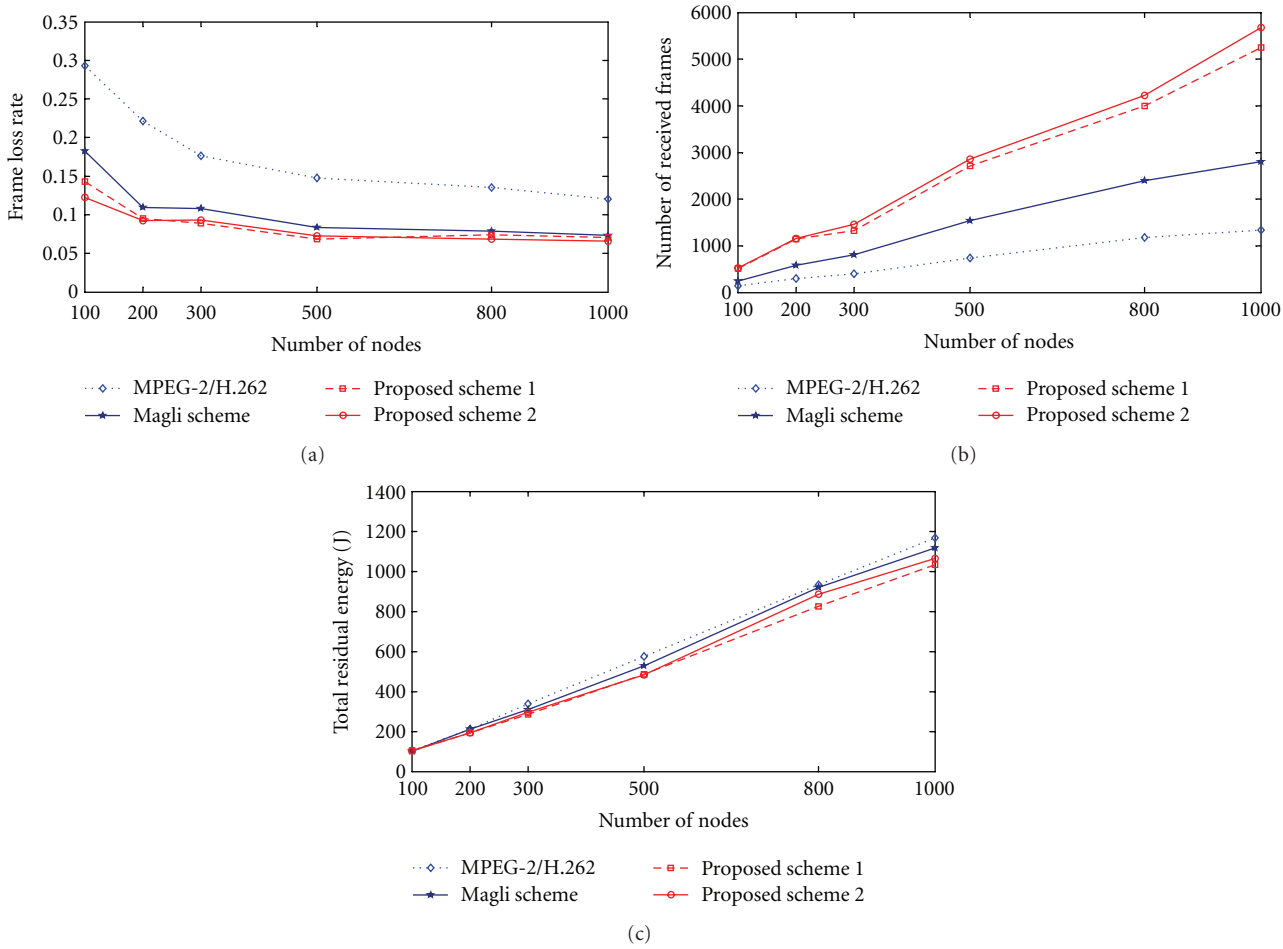

FIGURE 16: The results obtained for four schemes with QCIF *Akiyo* video in case of having channel error: (a) the average frame loss rate, (b) the number of received frames, and (c) the total residual energy.

the residual energy of the network after receiving 5000 frames at the base station. We conducted the simulation with 2000 sensor nodes, and the results are plotted in Figures 17 and 18. In Figures 17 and 18, the proposed schemes avoid that many sensor nodes run out of energy, and especially scheme 2 successfully makes the energy hole around the base station smaller than others.

4.3. Discussion. In this paper, we propose two schemes based on an edge detection technique. The proposed schemes achieve not only to improve the quality of decoding frames but also to balance the energy consumption of the nodes. However, there remains three issues in this paper.

First, we assumed that the backgrounds were not changing so quickly and we considered the scenes with small changes in the backgrounds and low motion of object. In the surveillance applications, the assumption will be acceptable. However, when the assumption is extended to other applications where backgrounds change quickly, the proposed algorithm will not be optimized, as shown in Figure 12. In the worst case, active region might be the same size of the original image. To solve the problem, the number of backgrounds should be large enough to remove noise when estimating motion regions and motion vectors. Since memories of sensors are limited, we will consider the tradeoff between the number of backgrounds and capacity of memory when applying for real environments in the future work.

Secondly, we assumed that there was only one object in the simulation. In the real monitoring environments, more than two objects might come into a frame. In this case, the most difficult problem is to determine motion region when these objects overlap each other. In the future work, we will therefore consider to solve the problem by using other techniques such as overlapping views technique [22].

Thirdly, we shared the edge detection task for a transforming node in the second proposed scheme but finding motion vectors task was still performed at a source/transforming node. The finding motion vectors task consumes more energy and memory [36, 37, 42]. Therefore, we will perform to distribute the task for multiple nodes to balance energy consumption on WSNs in the future work.

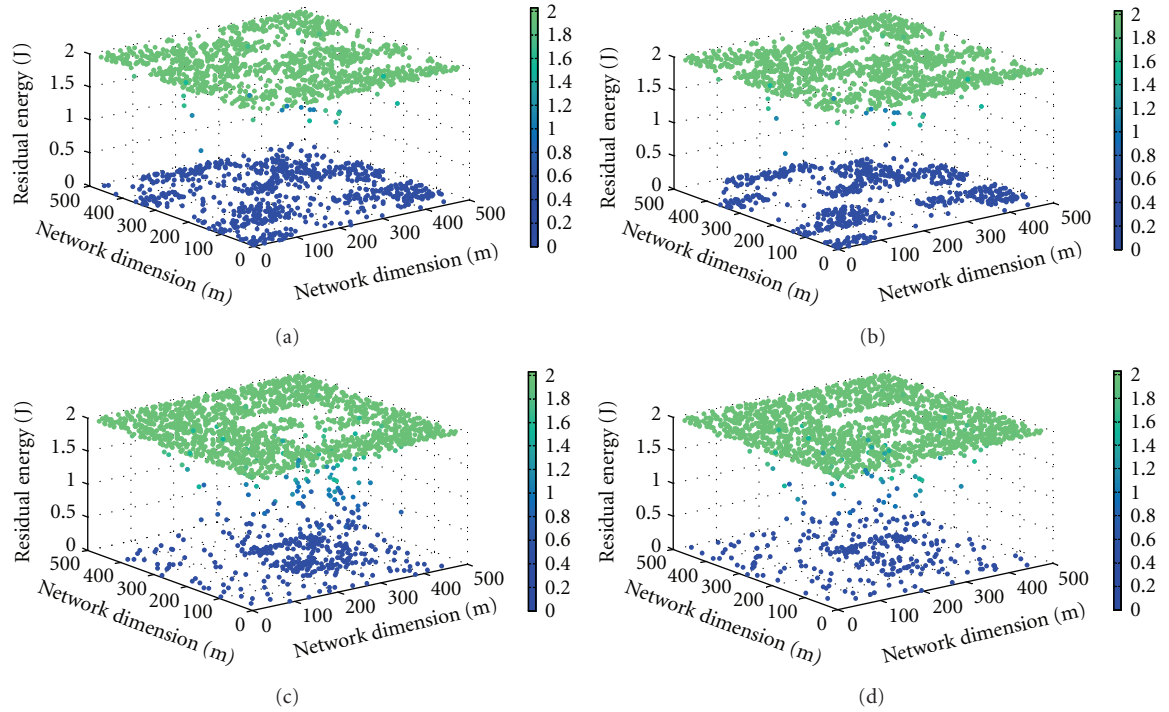

Figure 17: The obtained results after receiving 5000 frames at the base station with QCIF *Akiyo* video in case of not having channel error: (a) MPEG-2/H.262 scheme, (b) Magli scheme [20], (c) the proposed scheme 1, and (d) the proposed scheme 2.

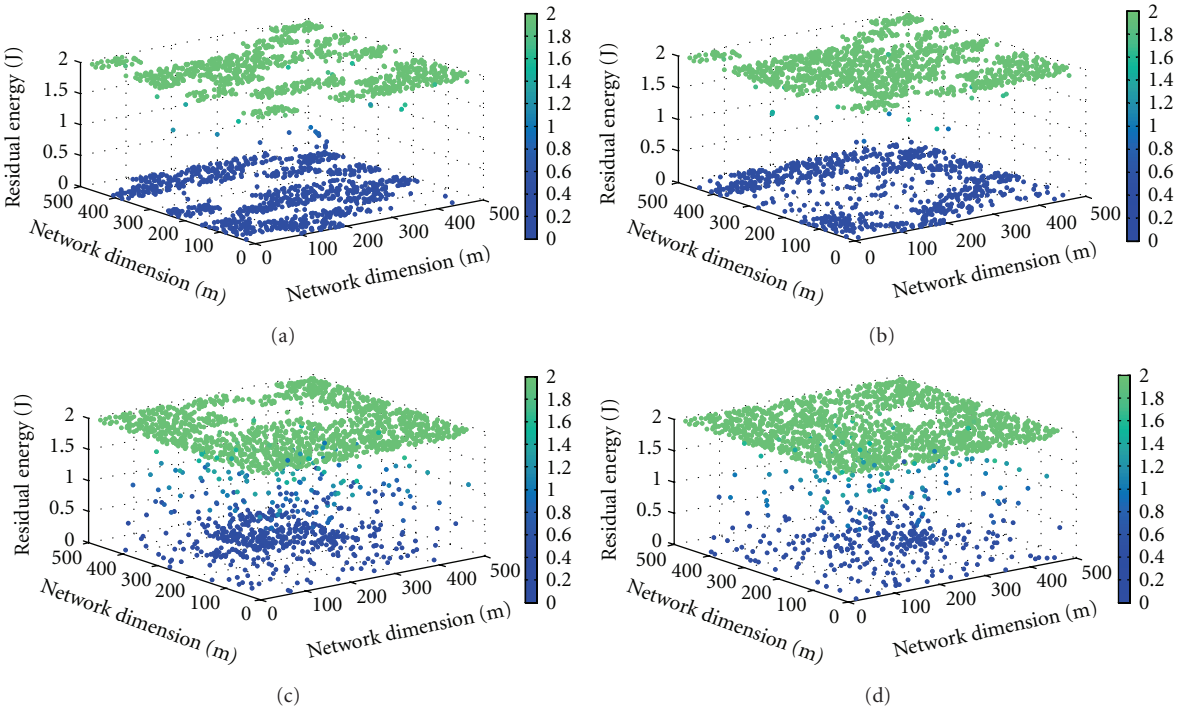

Figure 18: The obtained results after receiving 5000 frames at the base station with QCIF *Akiyo* video in case of having channel error: (a) MPEG-2/H.262 scheme, (b) Magli scheme [20], (c) the proposed scheme 1, and (d) the proposed scheme 2.

Figure 19: A general JPEG compression model.

5. Conclusion and Future Work

In this paper, we proposed two video compression schemes using an edge detection technique for balancing energy consumption in WVSNs. The proposed schemes solved three problems on WSNs, energy conservation, resource-constrained computation, and data processing. The energy conservation and data processing problems are solved by distributing the data processing tasks for multiple nodes from a source node to the cluster head in a cluster. As a result, the number of received data by the proposed schemes increases while the energy consumption is just the same for all schemes and energy among sensor nodes is balanced. For the resource-constrained computation problem, we use edge feature of image to find motion regions. The advantages of the technique are short execution time, low computational complexity, and low error rate [9].

In our simulation, because we use only I-frames and P-frames in video compression, the rate of compression is not yet optimized. Moreover, since we use H.262 encoder for intraframes, the quality and compression rate of encoding frames have not yet exploited. To solve the problem, the authors [23] used a backward channel to improve the quality of reference frames and compression rate of encoding frames while maintaining low complexity at the encoder. We therefore will utilize bidirectional frames (B-frames) and apply the features of H.264 encoder [45] for intraframes to improve the quality and compression rate in the future work.

Appendices

A. Coding Energy Model

The data processing models on WSNs have been investigated in many papers [40, 46–48]. To describe more details for compressing multimedia data, we use the simple JPEG model in [48], as shown Figure 19.

In the image/video compression algorithms, the data are often processed by 8×8 block. Consequently, we model the block-based energy consumption. Based on the model, the total energy consumption for compressing one data block, denoted as E_{cp}, is calculated as [48]

$$E_{cp} = E_{\mathrm{DCT}} + E_{\mathrm{Qt}} + E_Z + E_{\mathrm{RLE}} + E_{\mathrm{Huff}}, \tag{A.1}$$

where E_{DCT}, E_{Qt}, E_Z, E_{RLE}, and E_{Huff} are the consumed energy for transforming two-dimensional DCT (2D-DCT), quantizing, zigzag scan, RLE encoding, and Huffman encoding, respectively.

The 2D-DCT has been used in many papers, and it is often simply described as

$$\begin{aligned} F(k_1,k_2) &= \frac{X(k_1)X(k_2)}{\sqrt{2N}} \\ &\quad \times \sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} f(n_1,n_2)\cos\left(\frac{\pi}{N}\left(n_1+\frac{1}{2}\right)k_1\right) \\ &\quad \times \cos\left(\frac{\pi}{N}\left(n_2+\frac{1}{2}\right)k_2\right) \\ &= \frac{X(k_1)X(k_2)}{\sqrt{2N}} \\ &\quad \times \sum_{n_1=0}^{N-1}\left(\sum_{n_2=0}^{N-1} f(n_1,n_2)\cos\left(\frac{\pi}{N}\left(n_2+\frac{1}{2}\right)k_2\right)\right) \\ &\quad \times \cos\left(\frac{\pi}{N}\left(n_1+\frac{1}{2}\right)k_1\right), \end{aligned} \tag{A.2}$$

where $f(n_1, n_2)$ is the grey level of one pixel at position (n_1, n_2) in one block. X_{k_1} and X_{k_2} are equal to $1/\sqrt{2}$ if $k_1 = k_2 = 0$, and to 1 if $0 < k_1, k_2 < N$. The notations k_1 and k_2 are the discrete frequency variables such that

$$\begin{aligned} 0 \le k_1 < N, \\ 0 \le k_2 < N, \end{aligned} \tag{A.3}$$

where N is the block size. Equation (A.2) can be modeled by three $N \times N$ matrices as follows:

$$F_{(N\times N)} = A_{(N\times N)} I_{(N\times N)} A^T_{(N\times N)}, \tag{A.4}$$

where F is the matrix whose coefficients are $F(k_1, k_2)$, and the size of matrix is $N \times N$. I is the matrix of pixels of the original image, and A^T is the transpose of matrix A whose coefficients are expressed as the following equation,

$$A(k_1,k_2) = \begin{cases} \dfrac{1}{\sqrt{N}}, & \text{if } k_1 = 0, \\ \sqrt{\dfrac{2}{N}}\cos\left(\dfrac{\pi}{N}\left(k_2+\dfrac{1}{2}\right)k_1\right), & \text{otherwise.} \end{cases} \tag{A.5}$$

Each matrix $(N \times N)$ in (A.4) has N^2 coefficients, and each coefficient performs N multiplications and $(N-1)$ additions.

Therefore, total energy consumption for performing (A.2) is expressed as

$$E_{\mathrm{DCT}} = 2N^2(Ne_{\mathrm{mult}} + (N-1)e_{\mathrm{add}}), \tag{A.6}$$

where e_{mult} is the energy consumption per multiplication, and e_{add} represents the energy consumption per addition.

In the JPEG standard, the value of N is often 8, and thus the energy consumed for transforming DCT per block is expressed as

$$E_{\mathrm{DCT}} = 128(8e_{\mathrm{mult}} + 7e_{\mathrm{add}}). \tag{A.7}$$

Similarly, we can calculate energy consumption for quantization, zigzag scan, RLE encoding, and Huffman encoding steps. The more details of the steps can be seen in [48].

The energy consumption for performing quantization per block is calculated as follows:

$$E_{\mathrm{Qt}} = N^2(e_{\mathrm{div}} + e_r), \tag{A.8}$$

where e_{div} and e_r are the energy consumption for division and round instructions, respectively.

The energy consumption for performing zigzag scan per block is calculated as follows:

$$E_Z = (N^2 - 1)e_{\mathrm{sh}}, \tag{A.9}$$

where e_{sh} represents the energy consumption of shift process.

The energy consumption for encoding RLE per block is calculated as follows:

$$E_{\mathrm{RLE}} = (N^2-1)e_z + \Big((N^2-1) - \big(N_0 - N_{0\mathrm{seq}}\big)\Big)e_{wr} + \sum_{i=1}^{N_{0\mathrm{seq}}} e_{0\mathrm{seq}}(i),$$

$$e_{0\mathrm{seq}}(i) = \begin{cases} e_0, & \text{if } \big(i < N_{0\mathrm{seq}}\big) \vee (\text{Last\_component} \neq 0), \\ e_{wr}, & \text{otherwise.} \end{cases}$$

$$e_0 = (p_i + 1)(e_{wr} + e_{\mathrm{res}}) + (15p_i + q_i - 1)e_{\mathrm{inc}},$$

$$N_0 = \sum_{i=1}^{N_{0\mathrm{seq}}} \mathrm{length}(i),$$

$$P_i = \mathrm{length}(i) \operatorname{div} 16,$$

$$q_i = \mathrm{length}(i) \bmod 16, \tag{A.10}$$

where 0seq denotes the sequence of zeros, $N_{0\mathrm{seq}}$ is the number of 0seqs inside one block, and e_z is the consumed energy for checking whether an alternating current (AC) coefficient is null. The function length(i) returns the number of zeros in 0seq(i) (ith sequence of zeros). The notation e_{wr} is the dissipated energy for writing each 0seq of length $r < 16$ and the non-zero value X in the block denoted by (r, X). The notation e_{inc} is the dissipated energy for incrementing the counter of the number of zeros in each 0seq, and e_{res} is the consumed energy for resetting the counter.

Table 1: Energy parameters using in the simulation.

Parameter	Value
Initial energy	2 Joules
Energy of transceiver electron ($\varepsilon_{\mathrm{elec}}$)	50 nJ/bit
Energy for transmission in free-space model (ε_{fs})	0.01 nJ/bit/m^2
Energy for transmission in two-ray model (ε_{mp})	0.0000013 nJ/bit/m^4
Energy consumption for transforming DCT (e_{DCT})	20 nJ/bit
Energy consumption for performing zigzag (e_Z)	10 nJ/bit
Energy consumption for performing quantization (e_{Qt})	10 nJ/bit
Energy consumption for pre-processing (e_{Pre})	15 nJ/bit
Energy consumption for post-processing (e_{Post})	15 nJ/bit
Energy consumption for for coding (e_{Code})	90 nJ/bit
Energy consumption for performing edge detection (E_{Detect})	266846 nJ/frame
Energy consumption for finding motion region (e_{Mot})	1205 nJ/bit

The energy consumption for Huffman encoding per block is

$$E_{\mathrm{Huff}} = e_{\mathrm{Huff}}^{\mathrm{DC}} + e_{\mathrm{Huff}}^{\mathrm{AC}},$$

$$e_{\mathrm{Huff}}^{\mathrm{DC}} = e_1^{\mathrm{fetch}} + e_2^{\mathrm{fetch}} + e_d + e_{ws}, \tag{A.11}$$

$$e_{\mathrm{Huff}}^{\mathrm{AC}} = m\Big(e_3^{\mathrm{fetch}} + e_4^{\mathrm{fetch}} + e_{ws}\Big),$$

where m is the number of pairs (r, X) in previous RLE stage within the block, except the pairs that are special markers like (0, 0) or (15, 0), and e_{ws} is the energy required to write a stream of bits in the JPEG file. The notation e_1^{fetch} is the dissipated energy when we look in the category table for the representation of X, whereas e_2^{fetch} is the consumed energy when we look in the final step of Huffman encoding of the byte (r, X). The notation e_d is the energy required to compute the difference between two direct current (DC) coefficients (denoted as Diff$_j$). The notation e_3^{fetch} is the consumed energy when we look in the category table for the representation of Diff$_j$, and e_4^{fetch} is the dissipated energy when we look in DC Huffman table for the representation of the function Huff(cat(Diff$_j$)) written in the JPEG file for one block.

Based on [5, 7, 36, 40, 42], we calculate the energy for homogeneity edge detection and the energy for finding motion region. The more details of the steps can be seen in [5, 7, 36, 40, 42]. The notations and their values, which are used in the paper, are summarized in the Table 1.

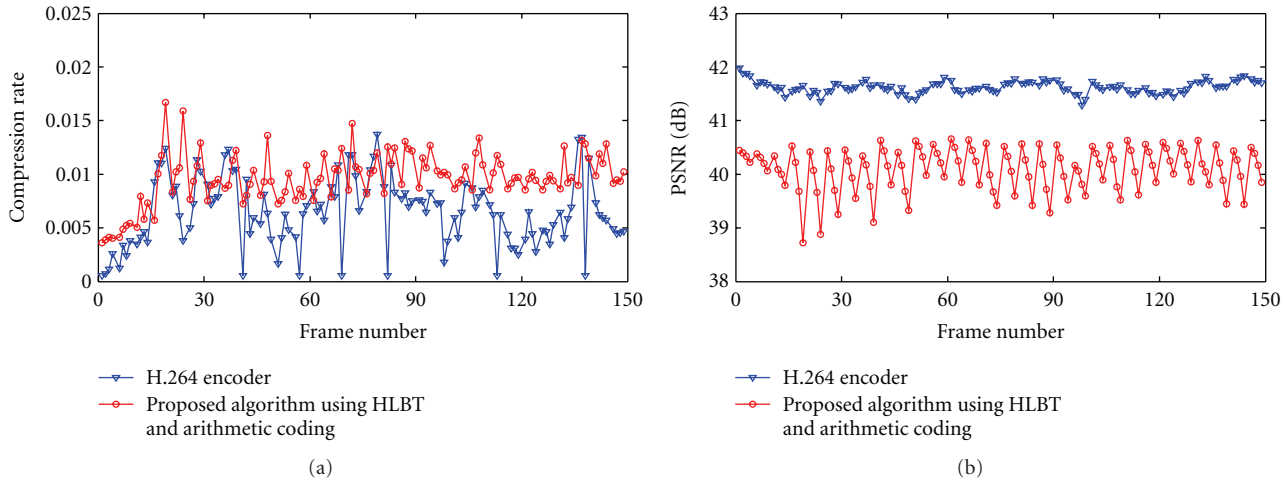

Figure 20: Simulation results with the *Akiyo* video: (a) the compression rate and (b) the quality of video.

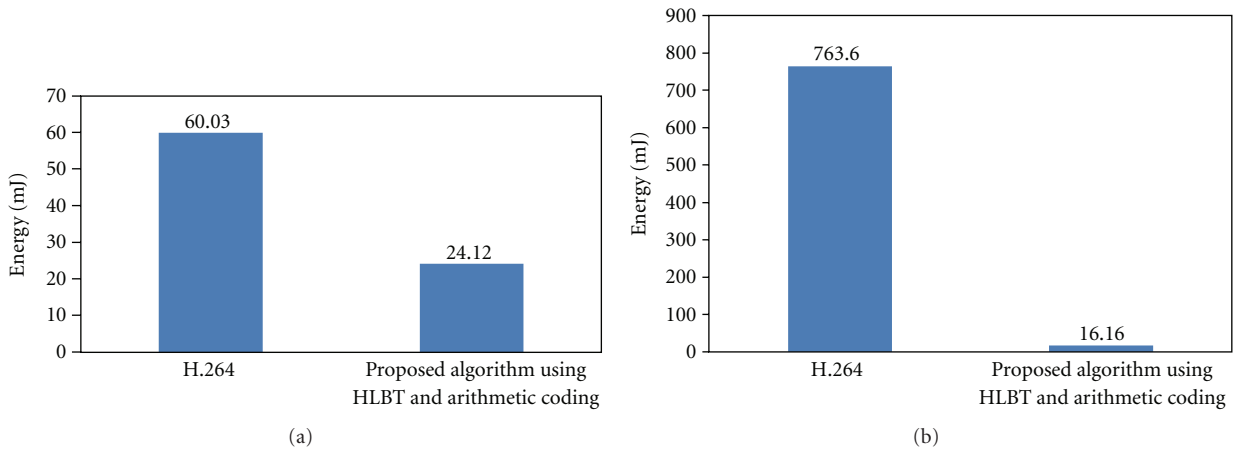

Figure 21: Comparing average encoding energy consumption among algorithms for *Akiyo* video with: (a) the intraframe and (b) the interframe (P-frame).

Table 2: Comparing computation complexity and coding gain with 0.3 bits/sample of four transformations with Gauss-Markov input and intersample autocorrelation coefficient $\rho \approx 0.95$ per block.

Transform (block size $N = 8$)	Number of multiplications	Number of additions	Coding gain (dB)
DCT	13	29	0
HLBT	16	42	0.65
LOT	22	54	0.40
LBT	23	54	1.05

B. Compare Quality and Energy Consumption of Proposed Algorithm with H.264/AVC

H.264/AVC is the new video coding standard of the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group. The main goal of the H.264 encoding is to improve both quality and compression rate of decoding video. In the standard, there are many highlighted features that enhance not only quality but also compression rate. However, it is difficult to implement the encoder for WVSNs because of its high complexity. In this section, we apply two typical features (lapped transform and arithmetic coding in H.264) for our proposed algorithm and compare with the H.264 standard.

B.1. Lapped Transform. Lapped transform (LT) is a technology that uses a preprocessing stage before implementing DCT to improve DCT [49, 50]. Therefore, we use an LT for intraframes in our proposed algorithm instead of DCT to improve the quality of decoding video. Among three types of lapped transform, namely lapped orthogonal transform (LOT), hierarchical lapped biorthogonal transform (HLBT), and lapped biorthogonal transform (LBT), we choose the HLBT because it is better than the LOT and LBT in three aspects: reduced blocking and ringing artifacts, lower computational complexity, and higher coding gain than LOT for low bit rate image coding applications as shown in

Table 2 [49, 50]. Therefore, the HLBT is the most suitable for implementing in WVSNs. The details of HLBT can seen in [49, 50].

B.2. Arithmetic Coding. The JPEG model is applied to compress video data in [4, 20, 51]. The advantage of the model is the simplicity of implementation. Nevertheless, since the Huffman coding is used in the JPEG model, we have to transfer the Huffman table with the compression data from source nodes to the base station in WVSNs. It thus consumes much energy of sensor nodes. Besides, the compression rate of decoding video in JPEG model is not high. To address the problem, we replace the Huffman coding by an arithmetic coding to improve an efficient compression [52, 53]. The main point of arithmetic coding is that each possible sequence is mapped to a unique number in $[0, 1)$. Therefore, efficient coding is improved. The details of the arithmetic coding can be seen in [52, 53].

B.3. Compare PSNR and Compression Rate. For H.264 encoder, we used a model in [4, 53] to estimate with our proposed algorithm. The results are shown in Figure 20. In Figure 20, the proposed algorithm, which uses HLBT and arithmetic coding, is competitive with H.264 encoder in terms quality (PSNR) and compression rate of decoding video.

B.4. Compare Encoding-Energy Consumption. In Figure 21, we evaluate the average energy consumption for encoding per frame at the source node between H.264 encoder and our proposed algorithm that uses HLBT and arithmetic coding. To evaluate the energy consumption for inter and intraframes of H.264 video compression, we use a model in [4, 53]. The results show that energy consumption for both intra and interframes in the proposed algorithm is much lower than that of H.264 encoder, especially in interframe (P-frame) as shown in Figure 21(b).

References

[1] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "Wireless multimedia sensor networks: applications and testbeds," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1588–1605, 2008.

[2] S. Soro and W. Heinzelman, "A survey of visual sensor networks," *Advances in Multimedia*, vol. 2009, Article ID 640386, 21 pages, 2009.

[3] A. Seema and M. Reisslein, "Towards efficient wireless video sensor networks: a survey of existing node architectures and proposal for a flexi-wvsnp design," *IEEE Communications Surveys and Tutorials*, vol. 13, no. 3, pp. 462–486, 2011.

[4] J. J. Ahmad, H. A. Khan, and S. A. Khayam, "Energy efficient video compression for wireless sensor networks," in *Proceedings of the 43rd Annual Conference on Information Sciences and Systems (CISS '09)*, pp. 629–634, March 2009.

[5] R. Nevatia, "A color edge detector and its use in scene segmentation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-7, no. 11, pp. 820–826, 1977.

[6] M. A. Ruzon and C. Tomasi, "Edge, junction, and corner detection using color distributions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1281–1295, 2001.

[7] S. Dutta and B. B. Chaudhuri, "A statistics and local homogeneity based color edge detection algorithm," in *Proceedings of the International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom '09)*, pp. 546–548, October 2009.

[8] A. Koschan and M. Abidi, "Detection and classification of edges in color images," *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 64–73, 2005.

[9] M. Setayesh, M. Zhang, and M. Johnston, "A new homogeneity-based approach to edge detection using PSO," in *Proceedings of the 24th International Conference Image and Vision Computing New Zealand (IVCNZ '09)*, pp. 231–236, November 2009.

[10] N. Senthilkumaran and R. Rajesh, "Edge detection techniques for image segmentation–a survey of soft computing approaches," *International Journal of Recent Trends in Engineering*, vol. 1, no. 2, pp. 250–254, 2009.

[11] R. Maini and H. Aggrawal, "Study and comparison of various image edge detection techniques," *International Journal of Image Processing*, vol. 3, no. 1, pp. 1–11, 2009.

[12] D. Ziou and S. Tabbone, "Edge detection techniques-an overview," *International Journal of Pattern Recognition and Image Analysis*, vol. 8, pp. 537–559, 1998.

[13] B. Tavli, K. Bicakci, R. Zilan, and J. M. Barcelo-Ordinas, "A survey of visual sensor network platforms," *Multimedia Tools and Applications*, vol. 60, no. 3, pp. 689–726, 2012.

[14] J. Hill, M. Horton, R. Kling, and L. Krishnamurthy, "The platforms enabling wireless sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 41–46, 2004.

[15] C. Chrysafis and A. Ortega, "Line-based, reduced memory, wavelet image compression," *IEEE Transactions on Image Processing*, vol. 9, no. 3, pp. 378–389, 2000.

[16] D. U. Lee, H. Kim, M. Rahimi, D. Estrin, and J. D. Villasenor, "Energy-efficient image compression for resource-constrained platforms," *IEEE Transactions on Image Processing*, vol. 18, no. 9, pp. 2100–2113, 2009.

[17] J. Oliver and M. P. Malumbres, "Low-complexity multiresolution image compression using wavelet lower trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 11, pp. 1437–1444, 2006.

[18] J. Oliver and M. P. Malumbres, "On the design of fast wavelet transform algorithms with low memory requirements," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 2, pp. 237–248, 2008.

[19] S. Rein and M. Reisslein, "Low-memory wavelet transforms for wireless sensor networks: a tutorial," *IEEE Communications Surveys and Tutorials*, vol. 13, no. 2, Article ID 5618529, pp. 291–307, 2011.

[20] E. Magli, M. Mancin, and L. Merello, "Low-complexity video compression for wireless sensor networks," in *Proceedings of the International Conference on Multimedia and Expo (ICME '03)*, vol. 3, pp. 585–588, July 2003.

[21] B.-S. Chow, "A limited resources-based approach to coding for wireless video sensor networks," *IEEE Sensors Journal*, vol. 9, no. 9, Article ID 5205184, pp. 1118–1124, 2009.

[22] C. Yeo and K. Ramchandran, "Robust distributed multiview video compression for wireless camera networks," *IEEE Transactions on Image Processing*, vol. 19, no. 4, pp. 995–1008, 2010.

[23] L. Liu, Z. Li, and E. J. Delp, "Efficient and low-complexity surveillance video compression using backward-channel aware Wyner-Ziv video coding," *IEEE Transactions on Circuits*

and Systems for Video Technology, vol. 19, no. 4, pp. 453–465, 2009.

[24] R. J. Clarke, "Image and video compression: a survey," *International Journal of Imaging Systems and Technology*, vol. 10, no. 1, pp. 20–32, 1999.

[25] I. W. Selesnick, R. G. Baraniuk, and N. G. Kingsbury, "The dual-tree complex wavelet transform," *IEEE Signal Processing Magazine*, vol. 22, no. 6, pp. 123–151, 2005.

[26] D. Zhao, Y. K. Chan, and W. Gao, "Low-complexity and low-memory entropy coder for image compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 10, pp. 1140–1145, 2001.

[27] I. W. Selesnick, "Hilbert transform pairs of wavelet bases," *IEEE Signal Processing Letters*, vol. 8, no. 6, pp. 170–173, 2001.

[28] S. Misra, M. Reisslein, and G. Xue, "A survey of multimedia streaming in wireless sensor networks," *IEEE Communications Surveys and Tutorials*, vol. 10, no. 4, pp. 18–39, 2008.

[29] B. Girod, A. M. Aaron, S. Rane, and D. Rebollo-Monedero, "Distributed video coding," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 71–83, 2005.

[30] Z. Xue, K. K. Loo, J. Cosmas, M. Tun, L. Feng, and P. Y. Yip, "Error-resilient scheme for wavelet video codec using automatic ROI detection and wyner-ziv coding over packet erasure channel," *IEEE Transactions on Broadcasting*, vol. 56, no. 4, pp. 481–493, 2010.

[31] O. Crave, C. Guillemot, B. Pesquet-Popescu, and C. Tillier, "Distributed temporal multiple description coding for robust video transmission," *Eurasip Journal on Wireless Communications and Networking*, vol. 2008, Article ID 183536, 2008.

[32] R. Puri, A. Majumdar, and K. Ramchandran, "PRISM: a video coding paradigm with motion estimation at the decoder," *IEEE Transactions on Image Processing*, vol. 16, no. 10, pp. 2436–2448, 2007.

[33] F. Pereira, L. Torres, C. Guillemot, T. Ebrahimi, R. Leonardi, and S. Klomp, "Distributed Video Coding: selecting the most promising application scenarios," *Signal Processing*, vol. 23, no. 5, pp. 339–352, 2008.

[34] S. K. Naik and C. A. Murthy, "Standardization of edge magnitude in color images," *IEEE Transactions on Image Processing*, vol. 15, no. 9, pp. 2588–2595, 2006.

[35] P. N. Tudor, "MPEG-2 video compression tutorial," in *Proceedings of IEE Electronics Division Colloquium on MPEG-2 What it is and What it isn't*, pp. 2/1–2/8, January 1995.

[36] X. Lu, E. Erkip, Y. Wang, and D. Goodman, "Power efficient multimedia communication over wireless channels," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 10, pp. 1738–1751, 2003.

[37] S. Yang, W. Wolf, and N. Vijaykrishnan, "Power and performance analysis of motion estimation based on hardware and software realizations," *IEEE Transactions on Computers*, vol. 54, no. 6, pp. 714–726, 2005.

[38] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[39] Q. Lu, W. Luo, and X. Ye, "Collaborative in-network processing of LT based image compression algorithm in WMSNs," in *Proceedings of the 1st International Workshop on Education Technology and Computer Science (ETCS '09)*, pp. 839–843, March 2009.

[40] Q. Lu, W. Luo, J. Wang, and B. Chen, "Low-complexity and energy efficient image compression scheme for wireless sensor networks," *Computer Networks*, vol. 52, no. 13, pp. 2594–2603, 2008.

[41] D. Varodayan, D. Chen, M. Flierl, and B. Girod, "Wyner-Ziv coding of video with unsupervised motion vector learning," *Signal Processing*, vol. 23, no. 5, pp. 369–378, 2008.

[42] T. H. Tran, H. M. Cho, and S. B. Cho, "Performance enhancement of motion estimation using SSE2 technology," *Proceedings of World Academy of Science, Engineering and Technology*, vol. 40, pp. 168–171, 2009.

[43] L. H. A. Correia, D. F. Macedo, D. A. C. Silva, A. L. Dos Santos, A. A. F. Loureiro, and J. M. S. Nogueira, "Transmission power control in MAC protocols for wireless sensor networks," in *Proceedings of the 8th ACM Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '05)*, pp. 282–289, October 2005.

[44] U. Datta and S. Kundu, "Packet level performance of a CDMA wireless sensor networks in presence of correlated interferers," in *Proceedings of the 4th International Conference on Computers and Devices for Communication (CODEC '09)*, pp. 1–4, December 2009.

[45] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[46] M. T. Sun and A. R. Reibman, *Compressed Video over Networks*, Marcel Dekker, New York, NY, USA, 2001.

[47] W. Lin, *Video Surveillance*, InTech, Rijeka, Croatia, 2011.

[48] A. Mammeri, A. Khoumsi, D. Ziou, and B. Hadjou, "Modeling and adapting JPEG to the energy requirements of VSN," in *Proceedings of 17th International Conference on Computer Communications and Networks (ICCCN '08)*, pp. 806–811, August 2008.

[49] H. S. Malvar, "Lapped biorthogonal transforms for transform coding with reduced blocking and ringing artifacts," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)*, pp. 2421–2424, April 1997.

[50] H. S. Malvar, "Biorthogonal and nonuniform lapped transforms for transform coding with reduced blocking and ringing artifacts," *IEEE Transactions on Signal Processing*, vol. 46, no. 4, pp. 1043–1053, 1998.

[51] H. Abid and S. Qaisar, "Distributed video coding for wireless visual sensor networks using low power Huffman coding," in *Proceedings of the 44th Annual Conference on Information Sciences and Systems (CISS '10)*, pp. 1–6, March 2010.

[52] N. Ling, "Expectations and challenges for next generation video compression," in *Proceedings of the 5th IEEE Conference on Industrial Electronics and Applications (ICIEA '10)*, pp. 2339–2344, June 2010.

[53] H.264/AVC. Reference software, jm11.0, http://iphome.hhi.de/suehring/tml/.