Performance Analysis of Bi-Level Image Compression Methods for Machine Vision Embedded Applications

Khursheed Khursheed, Muhammad Imran

Mid Sweden University, Sweden.

Abstract

Wireless Visual Sensor Network (WVSN) is an emerging field which combines an image sensor, an embedded computational unit, a wireless communication link, memory and energy resource. The individual Visual Sensor Nodes (VSNs) acquire image of the area of interest, perform some local processing on it and transmit the results using an embedded wireless transceiver. The processing of images at the VSN requires higher computing power and their transmission requires large bandwidth. Normally, the WVSNs are deployed in remote areas where the installation of wiring for power and data transmission is either not feasible or expensive. Due to the unavailability of continuous power, the energy budget in WVSN is limited. The limited energy budget requires that the processing at the VSNs and the communication to the server should consume as little energy as possible. The transmission of raw images wirelessly consumes a great deal of energy. Data compression methods can efficiently reduce the data and will thus be effective in reducing the communication energy consumption of the VSN. This paper explores seven well known bi-level image compression methods based on their processing complexity on the embedded platforms. The focus is to determine a compression method which can efficiently compress bi-level images and its processing complexity is suitable for the embedded platforms usually used for the implementation of the VSN. This paper is intended to be a resource for the researcher interested in using bi-level image compression methods in energy constrained real time embedded systems.

Keywords: Wireless Visual Sensor Network, Embedded Systems, Real Time Image Processing, Energy Consumption, Machine Vision.

1. Introduction

Wireless Visual Sensor Network (WVSN) is formed by deploying many Visual Sensor Nodes (VSNs) in the field. Typically, a VSN consists of an image sensor for capturing images of the environment, an embedded processing unit for on-board processing, memory for storage and a radio transceiver for communicating the results to the server. WVSNs are suitable for applications with a limited energy budget and are applied in remote areas where it is inconvenient to modify the location of the VSN or to frequently change the batteries.

The VSN should be capable of performing complex vision processing tasks such as morphology, labelling, object features extraction and image compression by using an embedded processing unit and must transmit the results wirelessly, but, unfortunately it possesses limited energy resources. Lower energy budget places stringent constraints on the type of the hardware components used for the implementation of the VSN. Typically, hardware components with low power characteristics are preferred for such applications. The energy consumption and bandwidth are major constraints in the remote applications of WVSN.

Typical examples of WVSN include the one explained in [1-3]. The authors in [1] developed a mote for wireless image sensor networks. They analysed the processing and memory limitations in current mote designs and developed a simple but powerful new platform. Their mote is based on a 32-bit ARM7 micro-controller operating at clock frequencies of up to 48 MHz and accessing up to 64 KB of on-chip RAM. They implied IEEE 802.15.4 standard for the wireless communication.

The authors in [2] presented CMUcam3 which is a low-cost, open source, embedded computer vision platform. Their hardware platform consists of a color CMOS camera, a frame buffer, a low cost 32-bit ARM7TDMI microcontroller, and an MMC memory card slot. The authors in [3] proposed and demonstrated a novel wireless camera network system, called CITRIC. The core component of this system is a hardware platform which is composed of a camera, a CPU (which supports up to 624 MHz clock speed), 16MB FLASH, and 64MB RAM. The design enables in-network processing of images to reduce communication energy consumption.

In our analysis, we have used three computing architectures based on AVR32, ARM and Intel which are NGW100 mkII, BeagleBoard-xM and a laptop machine. We explored the processing complexity of the seven compression methods on these three computing architectures. The NGW100 mkII kit uses the AT32AP7000 which has a 32-bit digital signal processor. The kit has 256 MB Random Access Memory (RAM) and 256 MB NAND flash. The AT32AP7000 operates at 150 MHz clock. The other computing platform is BeagleBoard-xM having ARM ® Cortex TM -A8 running at 1 GHz and 512MB RAM. The third computing platform is a laptop machine having Intel ® Core™2 Duo CPU with 1.86 GHz and 3 Giga Bytes (GB) RAM. The laptop machine is used both for cross compiling codes and libraries for the other two platform as well as for the analysis of the image compression methods.

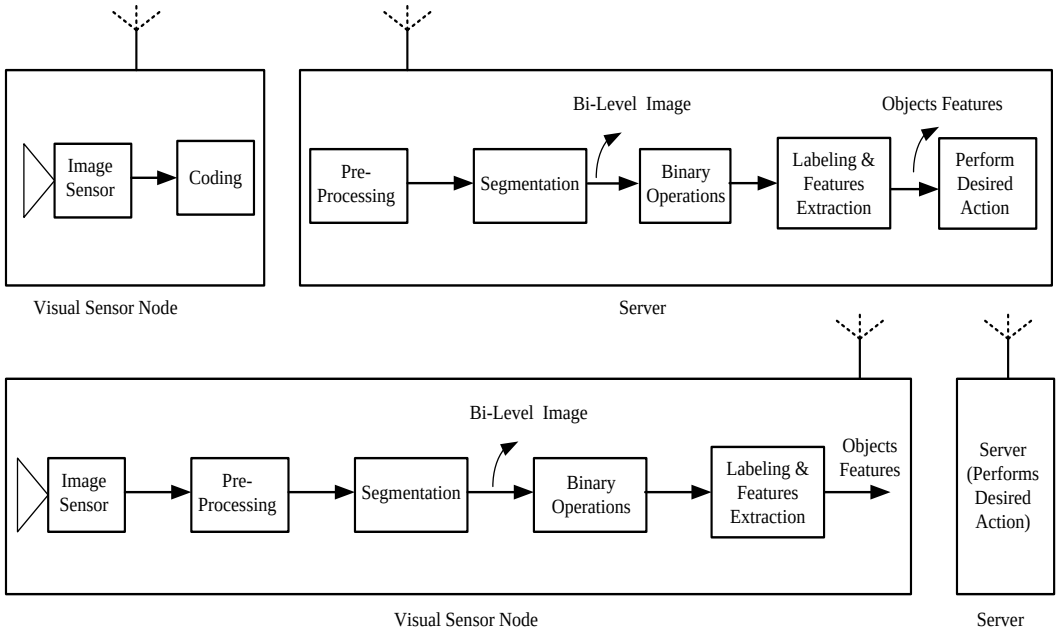

Fig. 1: The two extremes of image processing tasks in WVSN.

Both local processing and wireless communication consume a large portion of the total energy budget of the VSN. Transmitting the results from the VSN without local processing reduces the processing energy consumption but its consequence is higher communication energy consumption due to the transmission of large chunks of raw data. On the other hand, performing all the processing locally at the VSN and transmitting the final results reduces the communication energy consumption but the disadvantage of this is the higher processing energy consumption because of more processing at the VSN. These two extremes of processing are shown graphically in Fig. 1.

Our previous studies on intelligence partitioning between the VSN and server in [4, 5] have concluded that choosing a suitable intelligence partitioning strategy reduces the total energy consumption of the VSN. Transmitting the uncompressed images wirelessly from the VSN to the server quickly depletes its total energy. Communication energy consumption is heavily dependent on the amount of information that is being transmitted. Coding the binary image after pre-processing and segmentation is a good alternative in relation to achieving a general architecture for WVSN which is discussed in [6] and is shown in Fig. 2. Fig. 2 shows that the rest of the tasks such binary image processing operations, labelling and features extraction are performed at the server.

The amount of data after compression (Fig. 2) is fixed and is dependent on the compression standard used. Also the energy consumption of the VSN is dependent on the processing complexity of the implied compression standard. The aim of this work is the analysis of the binary image compression methods for their suitability to be used in the remote applications of WVSN.

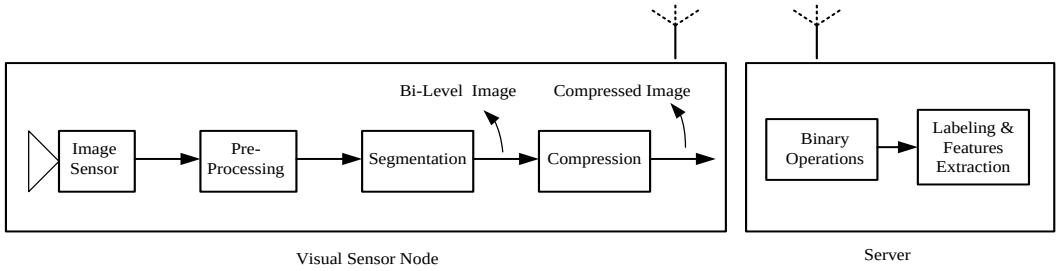

Fig. 2: Energy efficient architecture for WVSN.

Image compression techniques are broadly classified into two classes called lossless and lossy compression schemes. As the name suggests, in lossless (or information preserving) compression techniques, all the information regarding the image is preserved in the compressed data. In other words, the reconstructed image is exactly similar to the original image in every sense. Whereas in lossy (or irreversible) compression techniques, some image information is lost and the reconstructed image is only an approximation to the original image but is not exactly identical to it.

The compression provided by a lossless compression method is lower compared to that of lossy compression schemes, and thus lossy methods could be more favourable option for some image compression applications such as medical diagnostics. But the information preserving property of the image compression schemes is mandatory in many machine vision applications of WVSN. There is always a trade-off between lossy and lossless compression scheme. Its selection mainly depends on the requirements of application under consideration. In many applications of WVSN such as industrial automation, every object in the image is

equally important and must not be missed due to the lossy nature of the compression scheme. Our focus in this paper is all such applications which limit the scope of this investigation to lossless compression schemes only.

In a machine vision scenario, images contain few objects placed at random locations. Our focus in this work is on the analysis of lossless bi-level image compressing methods, based on various features of these images. It is possible for there to be significant variations in the size, shape and locations of the objects in the images. Thus, in order to analyse the image coding methods, a rich set of statistically generated images is a mandatory requirement.

Based on these statistically generated images, the desire is to see the behaviour of the considered compression methods from a number of different aspects. Specifically, we want to analyse the effect of increasing the number and the size of different shaped objects in the images on the processing complexity of the considered compression algorithms. Our aim is to determine a compression method which is resilient to the varying features of these objects in statistically generated images. Such a compression method must also be efficient in terms of compression ratio as well as processing complexity and hence will be suitable for energy constrained machine vision embedded systems.

The remainder of the paper is organized as follows. In section 2, the related work is provided and in section 3, the selected image compression methods are described. Section 4 presents the evaluation criterion for analyzing processing complexity of the bi-level image compression methods. This is followed in section 5 by a discussion of the performance evaluation. Section 6 provides the verification of the results based on the captured images of the machine vision environment. Finally, section 7 concludes the paper.

2. Related Work

Although the work on image compression and communication in wireless networks is both widespread and venerable, but the recent emergence of WVSN for large scale surveillance applications has, however, imposed new challenges because of the limitations on memory, processing complexity, communication bandwidth and energy consumption in WVSN. There are many bi-level image coding methods but to the best of our knowledge, there is no analysis of which of them present the best total energy consumption characteristic for energy constrained machine vision applications. For instance, the latest standard for compressing bi-level images from the Joint Bi-level Image Experts Group (JBIG) is JBIG2 [14]. However, the focus in the design of JBIG2 has been on the efficient storage of images, which has been fulfilled but, the main problem associated with JBIG2 for remote applications is its higher computational complexity.

Few researchers have compared the image compression methods, but the problem associated with all these comparisons is that their focus is only on the investigation of the compression ratio and speed on the personal computer. For instance, the comparison of international standards for lossless still image compression is investigated in [22]. They have thoroughly investigated the compression ratio of all the well-known compression methods available at that time and thus one problem of the comparison in [22] is the absence of the latest standard i.e. JBIG2. Another issue is the lack of compression method's

computational complexity consideration, which is the main factor to be considered for real time embedded systems. The energy consumption of the embedded implementation of an algorithm heavily depends on its processing complexity, which has not been investigated in [22].

In [23], both the compression ratio and processing complexity are investigated, but, the focus has not been on the energy consumption of the embedded implementation of the compression methods. In addition, they have investigated the efficiency of compression methods based on textual data, which is different from the images in a machine vision scenario.

Another study on the comparison of compression methods has been conducted in [24]. They have applied many compression standards to various medical images. They have compared both the compression ratio and the processing complexity of the compression standards, but, they have not evaluated the energy consumption of the embedded implementation. They have pointed out that the compression performance depends on the type of the images and the implication of this is that these results cannot be directly applied to machine vision applications because of the different types of images.

The compression performance of several lossless grayscale image compression algorithms is evaluated in [23] for textual data and in [28] for medical and natural images. So in literature, the analysis has been done for medical, natural and textual images. But the problem with all these analysis is the lack of consideration of the constraints of embedded platform. Their focus has been mainly on the compression rate or the processing speed at the personal computers. Also they have not evaluated the performance for images used in machine vision applications which is the scope of this work. In machine vision applications, the images are totally different from scanned textual and medical images and a thorough investigation of these kinds of images is required because the efficiency of bi-level image compression standards varies from image to image.

The H.264/AVC video coding standard is jointly developed by the ITU-T as recommendation H.264 and the ISO/IEC as international standard 14496-10 (MPEG-4 Part 10) Advanced Video Coding (AVC). It is the latest video compression standard which offers higher compression efficiency than the previous standards such as MPEG-l, MPEG-2, H.26l and H.263 standards. Unfortunately, the high level of compression of the MPEG-4 standard is achieved at the cost of high computational complexity. These high computational loads prevent in the remote applications of machine vision. The authors in [25] presented an analysis of the computational load for both the decoder and encoder programs of MPEG-4 on a standard personal computer. They concluded that the real-time issues of encoder programs are challenging.

The most computationally complex components of the video encoder (H.264, MPEG-4 part 10) are Motion Estimation (ME), Discrete Cosine Transform (DCT) coding and Mode Decision (MD). The processing complexity of these computationally complex components is too high for the computing platform used in VSN. Due to their higher processing complexity and lossy behaviour, the H.264 standard is not suitable for the energy constrained VSNs.

This paper investigates the most energy efficient bi-level image coding methods both in terms of processing complexity and energy consumption for energy constrained embedded computing platforms used in machine vision applications. We have studied seven well known bi-level compression methods and analysed

the computational complexity and energy consumption associated with each of these methods. The considered methods include JBIG2 [14], Gzip [17], Gzip_pack (for packed images), CCITT Group 3 [26], Group 4 [21], JPEG-LS [27] and one of the rectangular compression methods [12].

3. Evaluated Bi-Level Compression Methods

A number of lossless bi-level image compression algorithms exist. The well-known of these are the rectangular edge coding [7], rectangular coding (REC) [8], Modified Relative Element Address Designate (READ) coding [9], Ziv-Lempel algorithms [11], Efficient partitioning into rectangular regions [12], Arithmetic coding [13], LOCO-I (LOw COmplexity LOssless COmpression for Images) [10] etc. These algorithms are lossless in the sense that the compressed images retain all the information of the original data and an exact replica of the images can be reproduced at the receiver side. JBIG2 [14] is a lossless image compression standard which is based on a form of Arithmetic coding called MQ coder. The MQ coder used is an adaptive binary arithmetic coder, which is characterized by multiplication free approximation and a renormalization-driven update of the probability estimator, and bit stuffing introduced by Q-coder [15].

The Gzip (GNU zip) is a lossless universal standard used for document compression and is based on Ziv-Lempel algorithms [11]. We have used Gzip in two ways, one is Gzip and the other is Gzip_pack. This involves the same implementation, but, for Gzip we have used standard black and white images as the input (one byte per pixel). In relation to the Gzip_pack, we have firstly packed 8 pixels of the bi-level image into one byte and then compressed it using Gzip. The implementation for both Gzip and Gzip_pack is exactly the same (standard Gzip command in Linux) with only the input image data format being different.

The International Telegraph and Telephone Consultative Committee (CCITT) is a standard organization that has developed a series of communication protocols for the facsimile transmission of black-and-white images over telephone lines and data networks. These protocols are officially known as the CCITT T.4 [26] and T.6 standards [21] and are more commonly referred to as CCITT Group 3 and Group 4 (also known as Fax 3 and Fax 4) compressions, respectively. The JPEG-LS compression standard is based on the LOCO-I algorithm [10].

3.1 CCITT Group 3 (2D) and Group 4

CCITT Group 3 2-Dimensional (2D) and Group 4 are based on the Huffman coding and are almost similar except few differences. The Group 3 (2D) compresses k-1 lines (Typical value for k is 2 or 4) in 2 dimensional way and the kth line is coded in one dimensional way. On the other hand in Group 4 the image is always compressed 2 dimensionally. Another difference is that Group 3 (2D) inserts the end of line (EOL) after coding each line of the image for preventing propagation errors. All other aspects of these two methods are similar which are briefly explained below.

For each pixel of the image, in both Group 3 and Group 4 algorithms, there is a comparison operation for detecting changing picture element (transition from white to black or black to white pixels). Whenever the changing picture element is detected, a subtraction operation is performed to calculate the distance (in terms of number of pixels) from the previous changing picture element. If the distance is greater than 64 then it is repeatedly divided by 64 until the remainder is less than

64. Based on the distance and the remainder, there could be two to four memory accesses for accessing the Huffman codes. These codes are accessed and placed in the output buffer. Then the search for another changing picture element is started. In this way the whole image is coded. At the end, the output buffer along with the header information is written into the output image. Interested readers are referred to a detailed explanation of these standards in [26], [21].

3.2 Efficient Partitioning into Rectangular Regions

The authors in [12] proposed a bi-level image compression method. According to this method, the black regions of the image are partitioned into non-overlapping rectangular regions and then the locations of two vertices of each rectangle are encoded. These vertices can be used at the receiver side to fully reconstruct the bi-level image and, by this means, compression is achieved. This method is efficient for compressing bi-level images which only have rectangle shaped objects. The compression efficiency of this method is poor for images with irregular shaped objects such as curves [16].

3.3 The Gzip

The Gzip universal compression standard is based on LZ77 [11], which is a dictionary based lossless compression scheme and is usually used for text compression. The Gzip uses a sliding window of size *N*. *N* could be of any size, and in the implementation of [17], *N* is 32 Kilo Bytes (KB). A sliding window of 32K means that the compressor has a record of last 32K characters in the input file. When the next sequence of characters to be compressed, is identical to that which can be found within the sliding window, the sequence of characters is replaced by two numbers: a distance, representing how far back into the window the sequence starts, and a length, representing the number of characters for which the sequence is identical. It means that each new pixel of the image is compared to a pixel in the sliding window until a match is found. In the worst case there could be *N* comparison for each new byte (pixel). Following this method, the long sequences are replaced by only two binary numbers of a few bytes. In this manner, the compression is achieved.

3.4 The JBIG2

The compression algorithm employed in JBIG2 is arithmetic coding [13] which when supplied with accurate probabilities, provide optimal compression. The arithmetic coding is a form of variable length entropy coding usually implied in lossless data compression. In this method, a string of characters is represented using a fixed number of bits per character. In arithmetic encoding the frequently occurring characters are represented with fewer bits and the less-frequently used characters are represented with more bits. This procedure results in fewer bits for the overall representation of any string of characters. Arithmetic coding produces a single number for the entire message. The interested readers are referred to the details of arithmetic coding in [13] and that of JBIG2 in [14].

3.5 JPEG-LS

The JPEG-LS is a standard for lossless and near-lossless compression of continuous tone images. The algorithm at the core of JPEG-LS is the LOCO-I (Low Complexity Lossless Compression for Images). The JPEG-LS achieved compression ratios similar to those obtained with the state-of-the-art compression schemes based on arithmetic coding. Moreover, it is within a few percentage points of the best available compression ratios at a lower complexity level. Interested readers are referred to the details of JPEG-LS standard in [27] and that of LOCO-I in [10].

4. Evaluation Criteria

An important performance feature, which must be considered in analysing compression methods, is the processing complexity for both the compression and decompression processes. A compression algorithm is useless if its processing complexity introduces an intolerable delay in the image processing application. A higher processing complexity leads to higher energy consumption.

In remote applications of WVSN, the energy consumption is the main concern because of the limited energy resources. Thus, the processing complexity of a compression algorithm must be evaluated for the considered compression methods.

The performance of bi-level image compression algorithms is highly dependent on the transitions from the black to white and from white to black pixels in the image. In order to determine the performance of the compression algorithms under observation in this paper, we generated images of size 640x400 with random features of the objects in a black background. In one set of statistically generated images, we increased the number of objects while in another set of images, the number of objects remained fixed but we increased their sizes instead. Both of these sets of images contain objects with a variety of shapes such as circles, semi-circles, quarters of circles, ellipses, semi-ellipses, and quarters of ellipses, rectangles and curves. Our goal is to determine the trend in relation to the processing complexity on the embedded platform for the compression methods, relating to a variety of changes (fully random) in the input images such as the number, shapes and the sizes of the objects.

The compression rate of a compression method does not depend on the hardware or software architecture of the machine. On the other hand the processing complexity of a compression method is dependent on both the hardware and software architecture of the machine. The aim of our study is not to find the absolute processing complexity of a compression method on a specific machine, but instead we want to analyse the trend of the different compression methods. Generally speaking, it will help the readers to decide that which compression method is computationally efficient (or which one is computationally complex).

4.1 Analysing effect of increasing number of objects in the images

For each shape of white object, such as curves, circles, rectangles, ellipses etc., we generated 10 images using Matlab script and increased the number of objects by 10% in the successive images from Image1 to Image10. In total, for eight shapes of the objects we analysed 80 images.

Five sample images with one shape i.e. a quarter of an ellipse are shown in Fig 3. The randomness in the placement of the objects and the increase in the number of objects are obvious in Fig. 3 from (a) to (e).

Fig. 3: Images with randomly placed and increased number of objects from (a) to (e).

We analysed the processing complexity for eight different shaped objects in the images. For each shape, we have compressed 10 images, which have a 10 % increase in the number of objects in each successive image and presented the compression efficiency of the considered methods in [16]. It is concluded in [16] that the compression efficiency decreases with an increase in the number of objects in the images for all the compression methods. Additionally, the sensitivity is different for different compression methods in relation to the increase in the number of objects. We found that among the considered bi-level image compression methods, the Rectangular compression method is the most sensitive while the JBIG2 and CCITT Group 4 are the least sensitive to the increase in the number of objects in the images.

In the current work, we executed all the images with different shapes on the three computing platforms and determined the processing complexity of the compression methods. Some methods showed a high standard deviation in the processing complexity on the embedded platform in comparison to the others. The mean value and the standard deviation of the processing complexity for all the images with different shapes and an increasing number of objects are shown in Table 1 in section 5 for one of the computing platforms i.e. NGW100 mkII.

4.2 Analysing effect of increasing object's size in the images

For each shape of the white object such as curves, rectangles, circles, ellipses and rectangle, we generated 10 images using Matlab script and increased the sizes of the objects by 10%, in each successive image from Image1 to Image10. In total, we analysed 80 images for eight different shapes of the objects in the images.

Five sample images with one shape i.e. full circle are shown in Fig 4. The random placement and gradual increase in sizes of the objects are obvious in the images.

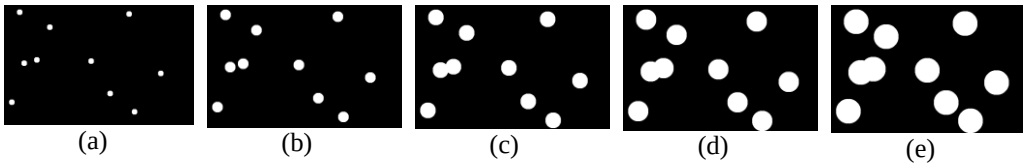

Fig. 4: Images with randomly placed and increased object's size from (a) to (e).

The compression ratio for these compression methods, in relation to the increase in the object size for each shape is presented in [16]. It is concluded in [16] that the compression ratio decreases as the size of the objects increases in the images for all the compression methods. Additionally, the sensitivity is different for different compression methods in relation to an increase in the size of the different shaped objects. For example, the sensitivity of the Rectangular method is lower for rectangular shaped objects and higher for curved shaped objects. This means that by increasing the sizes of the rectangles in an image, the size of the

compressed files does not increase for the Rectangular method. However, for the same method, the size of the output files increases tremendously for curved shaped objects. In terms of compressed file size, the conclusion was that, the Rectangular is the most sensitive while the JBIG2 and Group 4 are the least sensitive to an increase in the size of the objects in the images.
In the current work, all these sets of images are executed on the target embedded platform and the processing complexity together with the associated standard deviations for the compression methods is determined. For some of the methods the standard deviation, in relation to the processing complexity on the embedded platform is high, while for others it is low. The mean value and standard deviation for the processing complexity of all these sets of images are determined and are shown in Table 2 in Section 5 for one of the computing platforms i.e. NGW100 mkII.

5. Performance Evaluation

In this section, the processing complexity behaviour of the compression methods based on two sets of images having objects with a variety of features such as increasing numbers, sizes and various shapes are analysed on NGW100 mkII.
The algorithm for the Rectangular compression method explained in [12] is implemented using the C language and is cross compiled for the target embedded platform. For CCITT Group 3 and Group 4, the Libtiff library in [18] is cross compiled for NGW100 mkII. The Gzip compression is performed using the gzip command of embedded Linux running on NGW100 mkII. The JBIG2 uses the Leptonica image processing library for its input/output operations. The first action was to download the Leptonica image processing library from [19] and cross compile it for NGW100 mkII along with all the required functions of JBIG2.
All the cross compiled codes and respective libraries are then downloaded to the NGW100 mkII. The executable files of the compression methods are used to compress all the generated sets of the images on the embedded platform. For high accuracy reasons, the average execution time of compressing each image ten times is determined.

Table 1: Effect of increasing no. of objects on processing complexity and its standard deviation

Method	Mean Processing Complexity (t in ms)	Standard Deviation (σ_t ms)	Mean Sensitivity
Group 4	140	7	0
Group 3	140	2	0
JBIG2	348	5	0
Rectangular	554	111	0.04
Gzip	614	462	0.07
Gzip_pack	138	100	0.04
JPEG-LS	50	2	0

The mean sensitivity, mean processing complexity and its standard deviation for each of the considered compression methods are shown in Table 1 in relation to an increasing number of objects in the images. The larger values for the standard deviation of the processing complexity in Table 1 show that the Gzip (both) and Rectangular compression are highly dependent on the contents of the input image (i.e. number of objects). On the other hand, the JPEG-LS, CCITT Group 3, 4 and JBIG2 are resilient in terms of processing complexity to the contents of the input image. Table 1 also show that Group 3, 4, JBIG2 and JPEG-LS are the least sensitive (zero slope and low standard deviation) to the increasing number of

objects in the images. In addition, the mean processing complexity for the JPEG-LS, CCITT Group 3, 4 and gzip_pack on the target embedded platform are lower than the other compression methods.

Table 2 shows the mean and standard deviation for the processing complexity on the embedded platform based on an increase in the size of the objects in the images. Table 2 shows that the processing complexity for the Gzip, Gzip_pack and Rectangular compression methods are highly sensitive to an increase in the size of the objects in the input images. On the other hand, in terms of processing complexity, JPEG-LS, Group 3, 4 and JBIG2 are less sensitive to an increase in the size of the objects in the input image. Table 2 shows that Group 3, 4, JBIG2 and JPEG-LS are the least sensitive (zero slope and low standard deviation) to the increasing size of objects in the images. Additionally, the mean processing complexity for the JPEG-LS, Group 3, 4 and gzip_pack on the embedded platform are quite low.

Table 2: Effect of increasing size of objects on processing complexity and its standard deviation

Method	Mean Processing Complexity (t in ms)	Standard Deviation (σ_t ms)	Mean Sensitivity
Group 4	140	3	0
Group 3	140	2	0
JBIG2	351	8	0
Rectangular	600	215	0.03
Gzip	616	536	0.11
Gzip_pack	184	178	0.02
JPEG-LS	51	3	0

The compressed file size is different for the various compression methods and the transmission time of the radio transceiver is dependent on the size of data that is transmitted. In our previous work [4-6], the data transmission time for the IEEE802.15.4 radio transceiver is determined by using Equation (1). In our current work, IEEE802.15.4 is considered for the transmission of data from the VSN, thus, the same equation is used to calculate transmission time in Table 3.

$$T\_IEEE802.15.4 = (X+19)*0.000032 +0.000192 \quad \text{Equation (1)}$$

In Equation (1), X is the number of bytes transmitted and 19 is the overhead involved due to the inclusion of header information in each packet. Factor 0.000032 in equation (1) is the processing complexity of one packet while 0.000192 is the settling time of the transceiver.

Table 3: The Compression efficiency and processing complexity characteristics of the considered compression methods

Method	Mean Data Size (d in Bytes)	Standard Deviation in Data Size (σ_d)	Transmission Time (ms)	Transmission Energy (mJ)	Processing Time (ms)
Group 4	455	166	19	1.4	140
Group 3	2196	254	85	6.1	140
JBIG2	302	79	14	1.0	349
Rectangular	703	478	29	2.1	577
Gzip	2123	311	82	5.9	615
Gzip_pack	598	276	25	1.8	160
JPEG-LS	746	288	30	2.2	51

Fig. 5 shows the processing complexity vs. mean compressed file size for all the studied compression methods. The mean compressed file size is calculated by summing the compressed file sizes of the images, with different features such as increasing object size, increasing number of objects and different shapes, the sum

is then divided by the total number of images. The small circles in Fig. 5 show the mean compressed file size for each method on the horizontal axis and the average processing complexity (execution time on NGW100 mkII) on the vertical axis. The horizontal lines in Fig. 5 show the standard deviation in the compressed file size for each method.

It is clear from Fig. 5 that the mean compressed file sizes for the Gzip and CCITT Group 3 are high in comparison to all the other methods. It is also evident from the same figure that the processing complexity is high for Gzip, Rectangular and JBIG2 in comparison to all the other coding methods. It can thus be stated that JPEG-LS, Gzip_pack and CCITT Group 4 are good candidates to be used in energy constrained real time embedded systems. However, these trends of the compression methods must be verified on another embedded computing platform as well as for real captured images. Also the energy consumption characteristics of all the compression methods must also be analysed in order to provide a solid conclusion.

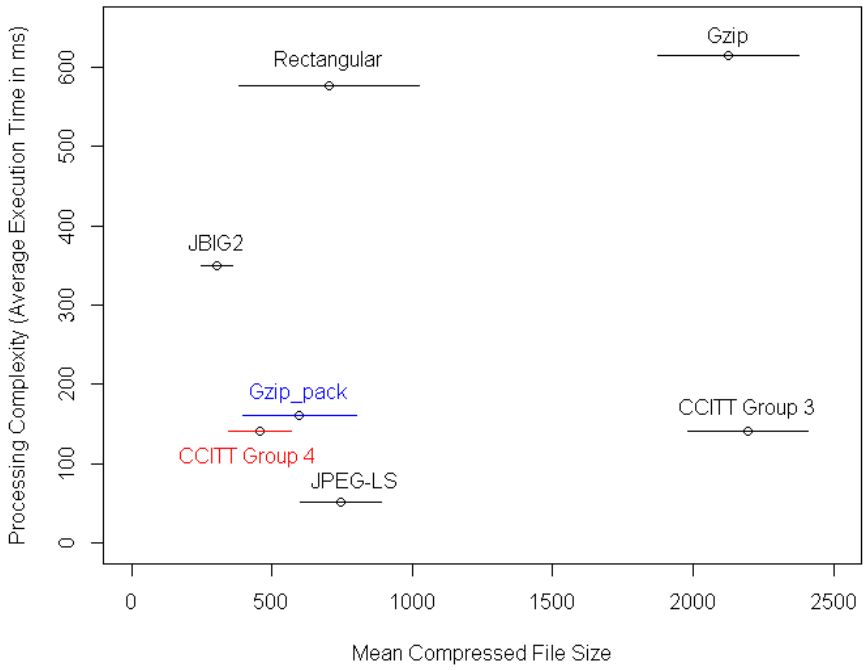

Fig. 5: Processing complexity vs. average files size of the compression methods.

6. Verification of the Performance Metrics

In order to verify the results of section 5, we captured 50 frames having varying number of objects of various shapes. The average compressed file size for both the statistically generated images and the captured images is shown in Fig. 6. It must be observed that the average compressed file size for the statistical and captured images is almost the same. This verifies one aspect of the developed statistical model i.e. the average compressed file size for both statistically generated and the real captured images is almost the same.

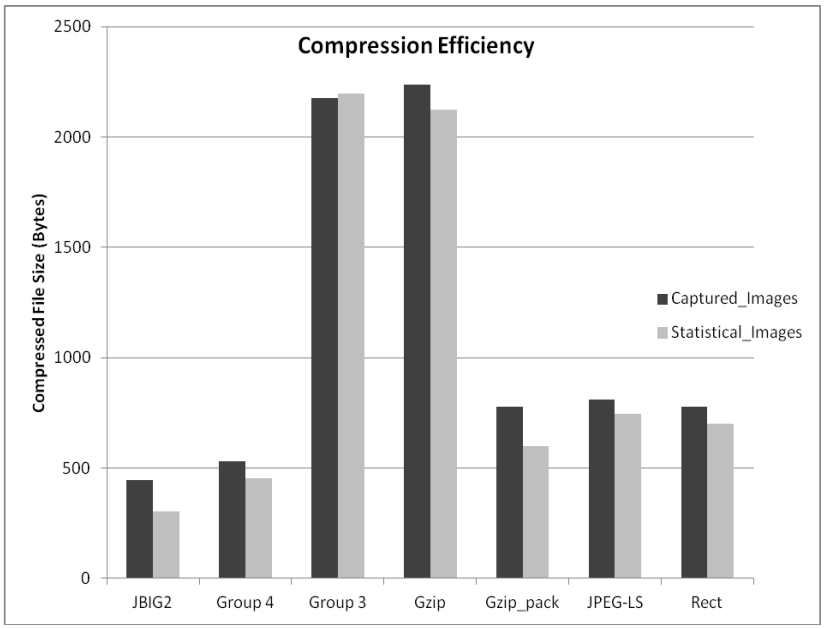

Fig. 6: The average compressed files size based on the captured and generated images.

Fig. 7 shows the comparison of the processing complexity for the captured and statistically generated images. The processing complexity is almost similar for both the captured and the statistically generated images. Fig. 7 shows that for the same computing architecture (i.e. AVR32), the processing complexity of all the compression methods for statistical and captured images is almost similar except two compression methods (The two methods are Gzip and Rectangular compression methods, which verifies their high standard deviation observed in the results of Section 5).

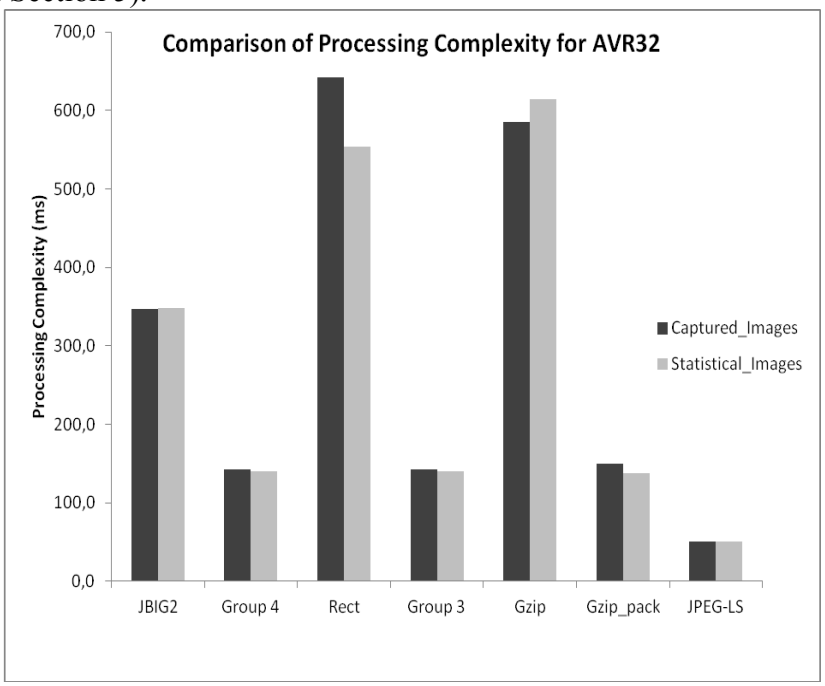

Fig. 7: The processing complexity of the captured and statistical images based on AVR32.

The processing complexity of the seven compression methods is determined by compressing the captured images using three different computing architectures i.e. Intel, ARM and AVR32. The detailed specifications of the computing platforms is explained in Section 1 i.e. Introduction. The processing complexity of all the

seven compression methods for the three computing platform based on captured images is shown in Fig. 8. The obvious result in Fig. 8 is that the processing complexity is lowest on the Intel machine and is the highest on the AVR32 embedded platform (The faster the computing system the lower the execution time). Another trend that must be observed in this graph is that the processing complexity of Gzip, Rectangular and JBIG2 compression methods is higher compared to other compression methods. The most important results in Fig. 8 is that the trend in the processing complexity of the compression methods is similar on the different computing architecture.

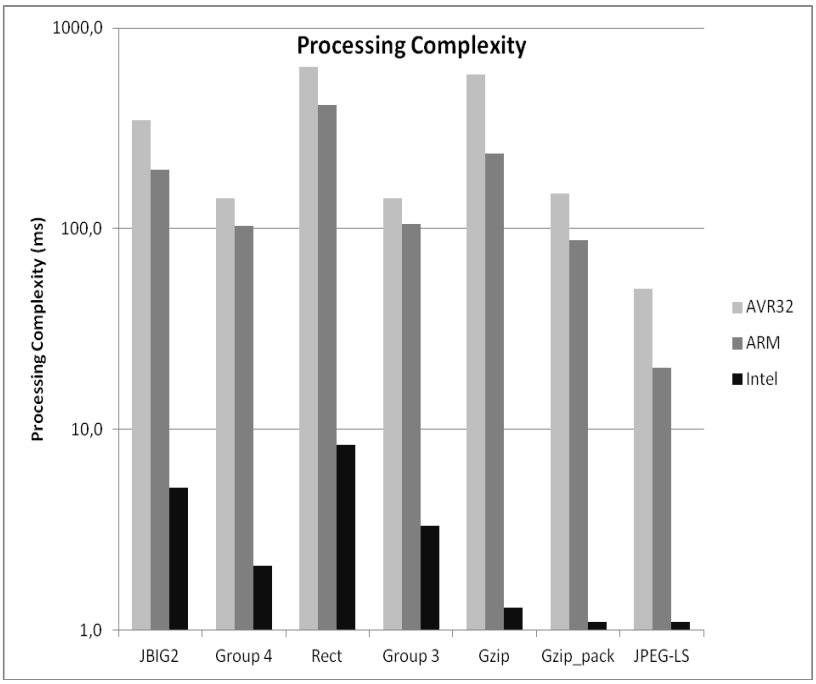

Fig. 8: The processing complexity of the computing platforms for the captured images.

The applied voltage and the current drawn are 9v and 110mA respectively for compressing the captured images on the NGW100 mkII. Similarly the applied voltage and the current drawn are 5v and 260mA respectively for compressing the captured images on the BeagleBoard-xM.

Table 4 shows the measured energy consumption of the seven compression methods for both of the embedded platforms i.e. NGW100 mkII and BeagleBoard-xM. In Table 4, the processing energy consumption is calculated by multiplying the applied voltage, the measured current and the measured average execution time of the compression methods on the embedded platforms.

Table 4: The processing and communication energy consumption of the compression methods

Method	Measured Parameters for NGW100 mkII				Measured Parameters for BeagleBoard-xM			
	Voltage (v)	Current (mA)	Execution Time (ms)	Measured Energy (mJ)	Voltage (v)	Current (mA)	Execution Time (ms)	Measured Energy(mJ)
Group 4	9	110	142	141	5	260	103	134
Group 3	9	110	142	141	5	260	105	137
JBIG2	9	110	347	344	5	260	197	256
Rectangular	9	110	642	636	5	260	512	666
Gzip	9	110	585	579	5	260	238	309
Gzip_pack	9	110	149	148	5	260	88	114
JPEG-LS	9	110	50	50	5	260	20	26

Fig. 9 shows the total energy consumption (Sum of processing and communication energy consumptions) for all the compression methods under consideration in this paper. It is clear from Fig. 9 that the total energy consumptions for the JPEG-LS, Gzip_pack and CCITT Group 4 are lower in comparison to all the other compression methods. The total energy consumptions of the Gzip and Rectangular compression methods are the highest among the considered compression methods. Based on these observations, it is possible to generalize that the CCITT Group 4, JPEG-LS and Gzip_pack are good candidates for energy constrained machine vision applications.

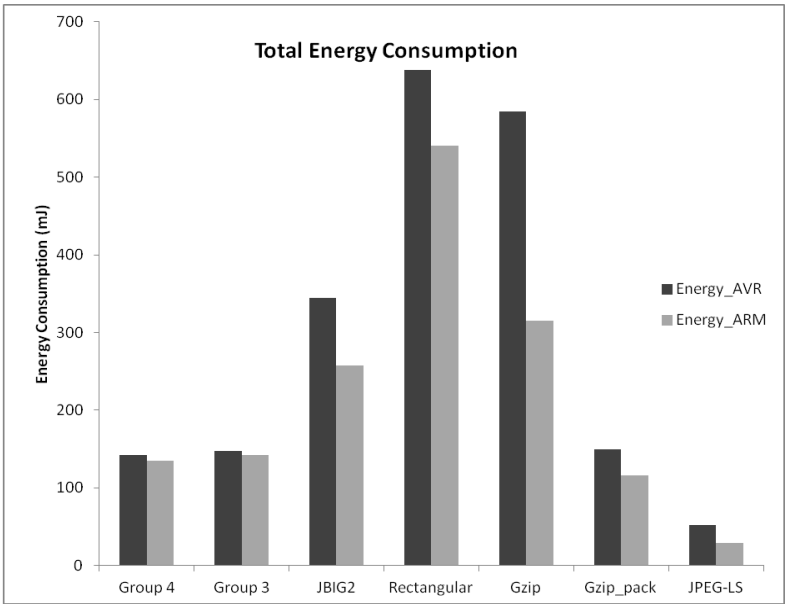

Fig. 9: Total energy consumption of the compression methods.

7. Conclusion

In this paper, we have analysed the effect of various features, such as the shapes, the sizes and the number of the objects in the bi-level images on the processing complexity of the seven well known bi-level image compression methods. We also used the measured total energy consumption for the comparison of the compression methods. The processing complexity of the JPEG-LS and CCITT Group 4 is highly resilient to both an increase in number and the size of different shaped objects within the input bi-level images. The Rectangular, Gzip and Gzip_pack compression methods showed higher standard deviation in their processing complexity for both an increase in size and number of different shaped objects in the input bi-level images. The JBIG2 is highly resilient to different variations of the features of an object and its compression efficiency is the highest among the compression methods. However, because of its high processing complexity, its total energy consumption is high. The consequence of higher energy consumption is reduced life time of the VSN, which is a characteristic that is not desirable for remote applications of wireless visual sensor networks. The hardware implementation of JBIG2 may be faster and this may result in lower total energy consumption. The measured energy consumptions of the embedded platform for JPEG-LS, CCITT Group 4 and gzip_pack are lower than all the other compression methods and hence these are the suitable candidates to be used in energy constrained machine vision embedded applications.

REFERENCES

[1]. Downes, I., Rad, L. B., Aghajan, H.: Development of a mote for wireless image sensor networks. In Proc. Cogn. Syst. Interact. Sensors (COGIS), Paris, France, Mar. 2006.

[2]. Rowe, A., Goode, A., Goel, D., Nourbakhsh, I.: CMUcam3: An Open Programmable Embedded Vision Sensor. Carnegie Mellon Robotics Institute Technical Report, RI-TR-07-13, May. 2007.

[3]. Chen, P., Ahammad, P., Boyer, C., Huang, H. S., Lin, L., Lobaton, E., Meingast, M., Oh, S., Wang, S., Yan, P., Yang, A. Y., Yeo, C., Chang, L.-C., Tygar, J., Sastry, S. S.: CITRIC, A low-bandwidth wireless camera network platform. In Proc. of the ACM/IEEE Int. Conf. Distributed Smart Cameras, 2008, pp. 1–10.

[4]. Khursheed, K., Imran, M., O'Nils, M., Lawal, N.: Exploration of Local and Central Processing for a Wireless Camera Based Sensor Node. IEEE Int'l. Conf. Signal Elec. Syst., Gliwice, Poland, Sept. 7-10, 2010.

[5]. Khursheed, K. Imran, M. Malik, A.W. O'Nils, M. Lawal, N. Benny, T.: Exploration of Tasks Partitioning Between Hardware Software and Locality for a Wireless Camera Based Vision Sensor Node. Proc. of the Int'l Symp. on Parallel Computing in Electrical Engineering (PARELEC'2011).

[6]. Imran, M., Khursheed, K., O'Nils, M., Lawal, N.: Exploration of Target Architecture for a Wireless Camera Based Sensor Node. 28th Norchip Conference 15-16 November 2010, Tampere, FINLAND.

[7]. Jayant, N. S., Noll, P.: Digital Coding of Waveforms: Englewood Cliffs, NJ: Prentice-Hall, 1984.

[8]. Aoki, M.: Rectangular region coding for binary image data compression. Pattern Recognition, Vol. 11, pp. 297-312, 1979.

[9]. Hunter, R., Robinson, A.: International digital facsimile coding standards. Proc, IEEE, Vol, 68, pp, 856867, July 1980.

[10]. Weinberger, M. J., Seroussi, G., Sapiro, G.: "LOCO-I: A low complexity, context-based, lossless image compression algorithm," in Proc. 1996 Data Compression Conf., Snowbird, UT, Mar. 1996, pp. 140–149.

[11]. Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. IEEE Trans. on Information Theory, vol. IT-23, no. 3, May 1977.

[12]. Mohamed, S. A., Fahmy, M. M.: Binary Image Compression Using Efficient Partitioning into Rectangular Regions. IEEE Trans. on Comm., Vol. 43, no. 5, May 1995.

[13]. Rissanen, J., Longdon, G. G.: Arithmetic coding. IBM J. Res. Develop Vol 23, pp 149-162, March 1979.

[14]. Ono, F., Rucklidge, W., Arps, R., Constantinescu, C.: JBIG2- The ultimate bi-level image coding standard. Proc. of the 2000 IEEE Intl. Conf. on Image Processing (ICIP), Vancouver, Canada, Sept. 2000.

[15]. Pennebaker, W. B., Mitchell, J. L., Langdon, G. G., Arps, R. B.: An overview of the basic principle of the Q-Coder adaptive binary arithmetic coder. IBM 1. Res. and Develop. Vol. 32 no. 6 page 717, November 1988.

[16]. Khursheed, K., Imran, M., Ahmad, N., O'Nils, M.: Selection of bi-level image compression method for reduction of communication energy in wireless visual sensor networks. Proc. SPIE 8437, 84370M (2012); http://dx.doi.org/10.1117/12.923716.

[17]. http://www.gzip.org/

[18]. http://www.libtiff.org/libtiff.html

[19]. http://tpgit.github.com/UnOfficialLeptDocs/leptonica/index.html

[20]. Papadonikolakis, M. E., Kakarountas, A. P.: Efficient high performance implementation of JPEG-LS encoder. J. Real Time Image Proc (2008) 3:303-310, DOI 10.1007/s11554-008-0088-7.

[21]. The link for T.6 i.e. CCITT Group 4: http://www.itu.int/rec/T-REC-T.6-198811-I

[22]. Arps, R.B., Truong, T.K.: Comparison of international standards for lossless still image compression. Proceedings of the IEEE, Vol 82, no 6, June 1994, 889 - 899.

[23]. Kodituwakku, S. R., Amarasinghe, U.S.: Comparison of Lossless Data Compression Algorithms for Text Data. Indian journal of computer science and engineering, 2010, Vol. 1 No. 4 416-425, ISSN : 0976-5166, Page No. 416-4125.

[24]. Kivijärvi, J., Ojala, T., Kaukoranta, T., Kuba, A., Nyul, L., Nevalainen, O.: A comparison of lossless compression methods for medical images. Computerized medical imaging graphics 1998; 22: 323-339.

[25]. Cavalli, F., Cucchiara, R., Piccardi, M., Prati, A.: Performance Analysis of MPEG-4 Decoder and Encoder. In Proc. of Int'l Symp. on Video/Image Processing and Multimedia Communications, Zadar, Croatia, 2002, 227-231.

[26]. The link for T.4 i.e. CCITT Group 3: http://www.itu.int/rec/T-REC-T.4-200307-I
[27]. Information Technology-Lossless and near-lossless compression of continuous-tone images-Baseline. International Telecommunication Union (ITU-T Recommendation T.87). ISO/IEC 14495-1, 1998.
[28]. Starosolski R.: Performance evaluation of lossless medical and natural continuous tone image compression algorithms. Proceedings of SPIE 2005; 5959:116–127.
[29]. Link to Images