Khursheed, K., Ahmad, N., Imran, M., & O'Nils, M. (2012). *Detecting and coding region of interests in bi-level images for data reduction in Wireless Visual Sensor Network.* Paper presented at the 1st International Workshop on Wireless Multimedia Sensor Networks (WMSN'12) , 8-10 Oct. 2012.
http://dx.doi.org/10.1109/WiMOB.2012.6379153

Detecting and Coding Region of Interests in Bi-Level Images for Data Reduction in Wireless Visual Sensor Network

Khursheed Khursheed, Naeem Ahmad, Muhammad Imran, Mattias O'Nils

Department of Information Technology and Media, Division of Electronics Design, Mid Sweden University
Sundsvall, Sweden

{khursheed.khursheed,naeem.ahmad,muhammad.imran, mattias.onils}@miun.se

Abstract—**Wireless Visual Sensor Network (WVSN) is formed by deploying many Visual Sensor Nodes (VSNs) in the field. The VSNs acquire images of the area of interest in the field, perform some local processing on these images and transmit the results using an embedded wireless transceiver. The energy consumption on transmitting the results wirelessly is correlated with the information amount that is being transmitted. The images acquired by the VSNs contain huge amount of data due to many kinds of redundancies in the images. Suitable bi-level image compression standards can efficiently reduce the information amount in images and will thus be effective in reducing the communication energy consumption in the WVSN. But compression capability of the bi-level image compression standards is limited to the underline compression algorithm. Further data reduction can be achieved by detecting Region of Interest (ROI) in the bi-level images and then coding these ROIs using bi-level image compression method. We explored the compression performance of the lossless ROI detection and coding method for various kinds of changes such as different shapes, locations and number of objects in the continuous set of frames. The CCITT Group 4, JBIG2 and Gzip are used for coding the detected ROIs. We concluded that CCITT Group 4 is a better choice for coding the ROIs in the Bi-level images because of its comparatively good compression performance and less computational complexity. This paper is intended to be a resource for the researchers interested in reducing the amount of data in the bi-level images for energy constrained WVSNs.**

Keywords: ROI coding, Image Coding, Wireless Visual Sensor Network, Energy Consumption.

I. INTRODUCTION

The work on image compression, Region of Interest (ROI) coding and communication over wireless networks is both widespread and venerable. However, the recent emergence of Wireless Visual Sensor (WVSN) for large scale surveillance applications has imposed new challenges because of the stringent limitations on memory, processing speed, bandwidth and energy consumption in a VSN.

In the literature many authors have focused on different implementation strategies for performing vision processing in the VSNs or the server (Figure 1). Some authors implemented a VSN which take the images of the field of view, compress the images and transmit them to the server for further processing [1]. In this case the communication cost is much higher because the VSN are not performing any vision processing and are transmitting the compressed images directly to the base station. The main processing unit at the VSN must also be in active mode for the communication of these raw compressed images and hence its power consumption must be considered.

On the other hand some authors proposed to perform all the vision processing tasks at the VSN and to transmit the object features to the server as the final results. In this case the communication cost is much lower but the computational cost is significant because the VSN is performing operations for a longer time. An example representing all the computation at the VSN is presented in [2] where the authors implemented a distributed vision processing system for human pose interpretation on a wireless smart camera network. They discussed that the motivation for employing distributed processing is to process the data in real-time and also provide scalability for developing more complex vision algorithms. By performing local processing at the smart camera they extracted critical joints of the subject in the scene in real time. The results obtained by multiple smart cameras are then transmitted through the wireless channel to a server for the reconstruction of the human pose.

Another such example is SensEye which is a multi-tier network of heterogeneous wireless nodes and cameras which aims at low power, low latency detection and low latency wakeup [3]. They implemented a surveillance application which performs advance image processing operations such as object detection, recognition and tracking.

Both local processing at the VSN and wireless communication consume significant portion of the total energy budget of the VSN. Transmitting the results from the VSN without local processing reduces the processing energy

This work is sponsored equally by Higher Education Commission Pakistan and Mid Sweden University Sweden.

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.

Previous studies on intelligence partitioning between the VSN and the server [4, 5] have concluded that choosing a suitable intelligence partitioning strategy reduces the total energy consumption of the VSN. Coding the binary image after pre-processing and segmentation is a good alternative in relation to achieving a general architecture for WVSN [6]. The architecture from [6] is shown in Figure 2. Based on this general architecture, we investigated the compression performance of various bi-level image compression standards in [7, 18] and concluded that JBIG2 [8], CCITT Group 4 [9] and Gzip_pack [10] provide good compression efficiency

The compression efficiency of change coding is explored in [17], which concluded that change coding provides better compression efficiency than image coding for up to 95 % changes in terms of number of object in the adjacent frames.

In current work, our aim is to explore the possibility of further data reduction (beyond the compression level provided by the compression of the bi-level image, see Figure 2) based on the detection and coding of ROIs in the bi-level image. The new method is shown in Figure 3.

In many applications such as meter reading, monitoring of magnetic particles in hydraulic system, monitoring of a habitat (e.g. monitoring of birds for preventing them to collide with windmills) in a specific area etc., the images contain few white objects in the black background. There is a possibility for further data reduction (beyond the compression level provided by image compression standard) if the Region of Interest (ROI) in the bi-level image is detected and compressed using suitable bi-level image compression method. Hence, by reducing the information amount that needs to be transmitted using ROI detection and coding method, the communication energy consumption of the wireless application will be reduced (in the rest of the paper sometime we will use the term ROI coding and by this we mean the detection and compression of the ROIs in the images).

The focus of this work is to design a computationally efficient lossless ROI detection and coding method for compressing bi-level images (Less complex ROI coding method is needed for wireless applications with low energy budget). We want to extract ROIs in the bi-level image and then compress these ROIs using a suitable bi-level image coding method. The compressed ROIs along with the information about their locations in the image (in the form of run length codes) can be transmitted wirelessly to the server. These compressed ROIs and the run length codes can then be used at the server to reconstruct the original bi-level image in a lossless manner.

The aim is to explore the compression efficiency of the ROI detection and coding method for different kinds of variations in the images. The possible variations in the bi-level images are the different shapes, locations and number of white objects in the images.

The intended application area for this work is Wireless Visual Sensor Network (WVSN), which can be used to monitor a specific phenomenon in the environment. WVSNs are suitable for applications with a limited energy budget and are applied in remote areas where it is inconvenient to modify the locations of the VSNs or to frequently change the batteries. Due to wireless nature of the application, the energy consumption and the bandwidth are the major constraints in WVSNs. By designing a simplified ROI detection and coding method for such applications, the energy consumption can be reduced.

The remainder of the paper is organized as follows. In section II, the related work is provided. Section III presents the idea of lossless ROI detection and coding method. This is followed by the results in section IV. The discussion of the results is provided in section V. Finally, section VI concludes the paper.

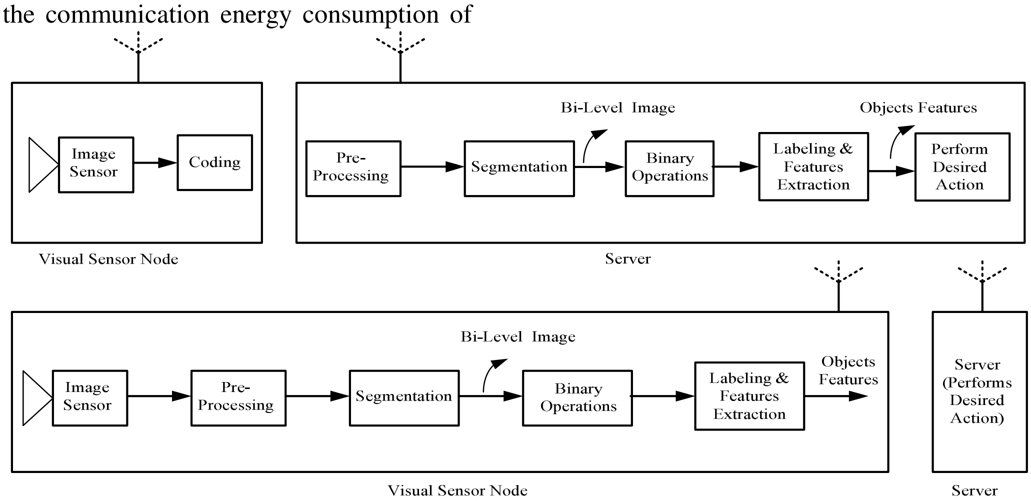

Figure 1. The two extremes of image processing tasks in WVSN.

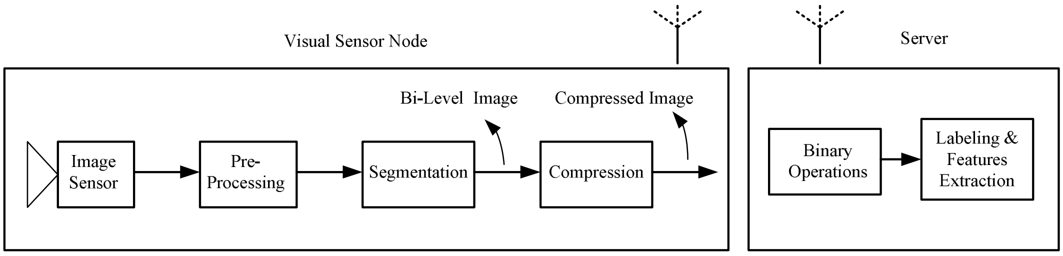

Figure 2. Energy efficient architecture for WVSN.

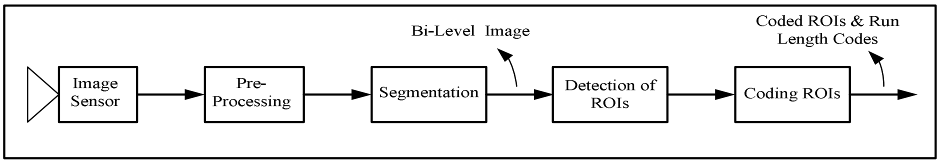

Figure 3. ROI detection and coding method.

II. Related Work

In this section, we provide the literature review of the state of the art in the ROI coding. ROI coding is applied in applications where only certain regions of the image are important for the end user. ROI coding has been extensively explored for medical images, where the practitioners are only interested in certain medically important areas of the images. Usually lossless image compression is required for a small part (ROI) of the image where the real interest of the end user (medical practitioner) is located. The rest of the image is needed only in a contextual terms and can thus be compressed in a lossy manner.

An early effort on progressive image encoding has been done in [11]. Using progressive encoding, they achieved different quality for different parts of the image. Using this method an arbitrary ROI in any image can be encoded progressively up to lossless.

A fast and efficient image compression algorithms based on set partitioning in hierarchical trees (SPIHT) has been proposed in [12]. This algorithm is based on the principle of partial ordering by magnitude with a set partitioning sorting algorithm, ordered bit plane transmission and exploitation of self-similarity across different scales of an image's wavelet transform. SPIHT is a powerful image compression algorithm that produces an embedded bit stream from which the image can be reconstructed at various bit rates.

Three mechanisms are available in the well known JPEG2000 compression standard for compressing different parts of an image with different spatial qualities i.e. tiling, code block selection and coefficient scaling. These three methods have been best described in [13].

The adaptive SPIHT [14] compression method can be used to compress different parts of an image with different compression algorithms providing different qualities for the ROIs and the background image. Selective compression is done by performing JPEG2000 on the ROIs and SPIHT on the rest of the image. The compression process becomes energy efficient by performing energy efficient SPIHT on the non-ROI parts of the image.

The problem with all these ROI compression methods is their high computational complexity which will end up in high processing energy consumption. Our aim is to propose an ROI coding method which is computationally less complex, so that the overall energy consumption of each of the VSN is reduced.

In certain machine vision applications the gray scale image can be segmented into bi-level images. The ROIs in these bi-level images can be of any arbitrary shapes. There is a need for a simplified ROI detection and coding method for bi-level images.

III. The ROI Detection and Coding Method

The architecture for the ROI detection and coding method is explained in Figure 4. The ROI detection algorithm determines the ROI_Image by removing the complete black rows and columns (non-ROI part) of the bi-level image. It also determines the location of the ROIs in the image and represents it in the form of run length codes. Suitable bi-level image compression algorithm is used to compress the ROI_Image. The compressed ROI image and the run length codes can be used to reconstruct the original image. The detection of the ROIs and the run length codes is explained in detail in sub section *A*.

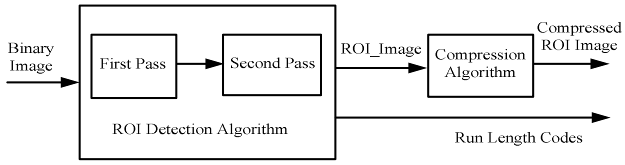

Figure 4. Architecture for ROI coding.

A. Detection of ROIs in Bi-Level Images

Figure 5 and Figure 6 show the first and second pass of detecting rectangle/square shaped ROIs in the bi-level image respectively. The run length codes required for the reconstruction purpose are also shown in the respective figures.

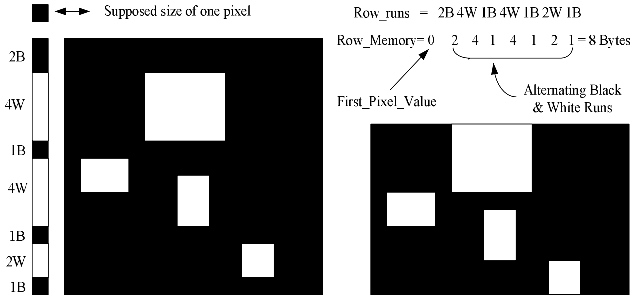

Figure 5. The first pass of the rectangle/square shaped ROIs detection process.

The supposed pixel size for both Figure 5 and Figure 6 is shown at the top-left corner of Figure 5. In the first pass of the ROI detection algorithm, black and white runs of the rows of the image is determined. The adjacent rows having no white pixels are termed as black rows and are not copied to the ROI_first_pass image. The adjacent rows having one or more white pixels are termed as white rows and are copied to ROI_first_pass image.

The value of the first pixel of the image is stored at the first place in the Row_Memory as is shown in the top-right corner of Figure 5. Then all the elements of the Row_runs are saved in the Row_Memory. A similar procedure is applied to the column of the processed image in the second pass of the ROI detection algorithm.

The image after the first pass of the ROI detection algorithm is shown in the right part of Figure 5, where only the rows having white objects are present. All the columns of ROI_first_pass image having no white pixels are not copied to ROI_second_pass image whereas all the columns of ROI_first_pass image having one or more white pixels are copied to ROI_second_pass image. The resultant image is shown in the right part of Figure 6.

Only those rows and columns of the original image having white objects are present in the final ROI image (ROI_Image). This final image is then compressed with the desired compression algorithm. The final compressed ROI image and the runs of the rows and the columns are transferred to the server.

For reconstruction, the dimension of the original image is used to generate a zero array (We used Matlab for the reconstruction process). The first element of the Col_Memory shows the color information of its second element. The rest of the elements of the Col_Memory are the alternating black and white runs. If the color of the second element of Col_Memory is black (based on the value of its first element) then the color of its third element is white (alternating black and white runs) and so on.

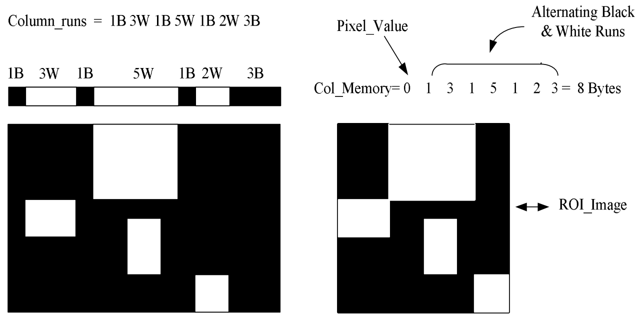

Figure 6. The second pass of the rectangle/square shaped ROIs detection process.

In the ROI detection process, in the first pass the rows are processed while the columns are processed in the second pass. On the other hand, in the reconstruction process, the columns are processed first and then the rows are processed.

For every white element of Col_Memory, respective number of columns of the ROI Image is copied to the newly generated zeros array. Then number of columns equal to the next black element of the Col_Memory are left unprocessed in the zeros array. This procedure is repeated for all the columns of the image. In a similar fashion, the Row_Memory and the processed zero array are used to fully reconstruct the original image in a lossless manner.

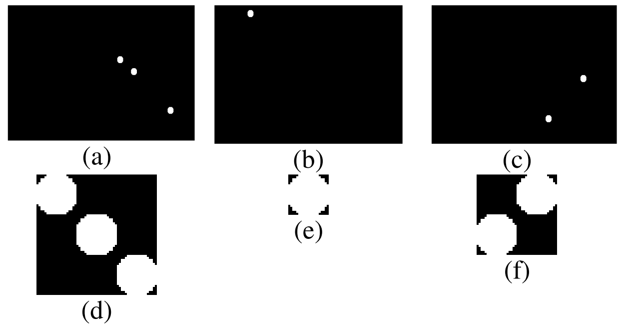

Figure 7. Images with circle shaped objects and the detected ROIs.

In order to determine the performance of the ROI coding, we have generated a sequence of 50 frames of large and small sizes (using a Matlab script) with random features of the objects in a black background. In the analysis, we used objects with various shapes such as circles, semi-circles, quarter of circles, ellipses, semi-ellipses, quarter of ellipses, rectangles, lines and curves.

Figure 7 (a)-(c) show the frames with varying number of circle shaped objects. The detected circle shaped ROIs (Figure 7 (d)-(f)) have exactly the same size as is in the original frames (Figure 7 (a)-(c)) but look large due to more space available.

It must be observed that all the rows and columns in Figure 7 (a)-(c) without any white pixel/objects are removed and hence are not present in the detected ROIs in Figure 7 (d)-(f). Only the circle shaped white objects at their corresponding locations are present in Figure 7 (d)-(f). The compression efficiency of the compression standards for compressing such frames and ROIs is shown in the results section.

B. *Analysis of the run length codes*

In this sub section we showed the analysis of the representation of the run length codes. Any arbitrary number of bits can be used to represent the run length codes, but its effect on the final compressed bit stream needs to be analysed. Selecting too few or too many bits for the representation of run length codes is not a good idea. Careful analysis needs to be done. Two different sizes of the frames i.e. large frames (3000X2000 i.e. 2000 rows, 3000 columns) and small frames (640X400 i.e. 400 rows, 640 columns) having varying number of objects from 0 to 20 are considered in this analysis.

Table 1 shows the number of bytes needed for the representations of the run length codes for small and large frames with varying number of objects. We analysed various number of bits such as 5, 6, up to 12 for the representation of the run length codes. The aim is to find the most appropriate number of bits for the representation of the run length codes.

It must be observed in Table 1 that the number of bytes of the run length codes is highly dependent on the number of bits used for its representation. If we use 5 bits for the representation of the run length codes then higher number of bytes is needed. The average number of bytes is the lowest for the case where 7 bits (on average **17** bytes are needed) and 9 bits (10 bits is also a good option) are used for the representation of the run length codes for small and large frames respectively (Table 1).

The representation (bit width) for the run length codes must be general and we have to choose the one which is suitable for both large and small frames. Choosing 9 bits for the representation of run length codes is the appropriate for both large and small frames (Table 1). The run length codes using 9 bits representation are added to the compressed file size of ROIs and the results are shown in the results section.

TABLE 1: ANALYSIS OF THE NUMBER OF BITS FOR THE REPRESENTATION OF RUN LENGTH CODES.

Frame Size	**No. of Objects**	**Bits = 5**	**Bits = 6**	**Bits = 7**	**Bits = 8**	**Bits = 9**	**Bits = 10**	**Bits = 11**	**Bits = 12**
Small	0 to 4	23	17	15	13	13	14	16	17
	5 to 10	26	20	19	21	23	24	28	31
	Average	25	19	**17**	18	19	20	23	25
Large	0 to 4	106	65	42	30	24	21	21	23
	5 to 10	116	75	56	49	48	52	56	68
	Average	112	71	50	41	**38**	**38**	41	48

IV. RESULTS

Libtiff library [15] is compiled and used for compressing images with Group 4. The Gzip compression is performed using the gzip command of the Ubuntu operating system. 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 [16] and compile it, after which all the required functions of JBIG2 are compiled to create the execution file for JBIG2.

ROIs of different shapes such as full circles, semi-circles, quarter of circles, full ellipses, semi-ellipses, quarter of ellipses, rectangles, lines and curves are extracted from the large and small frames. The small and large frames and the extracted ROIs are compressed using the three considered compression methods. The results are shown graphically for one shape i.e. semi-ellipse in this section.

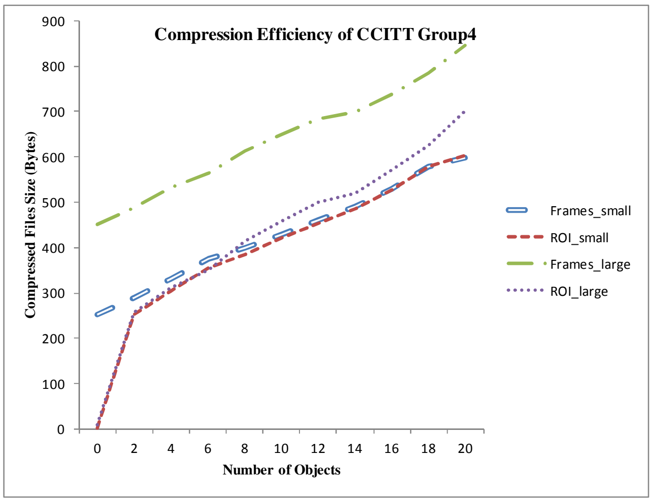

Figure 8. Performance of Group4 for ROI detection and coding method.

In order to get enough statistics, 50 frames for every case of the objects (i.e. 0, 2, 4… 20 objects) are compressed and their average compressed file size is determined. Vertical axis in Figure 8 shows the average value of the 50 compressed frames against every index of the horizontal axis (for CCITT Group 4). The top most curve in Figure 8 shows that the compressed file size for the large frames is high.

The interesting result in Figure 8 is the difference between the compressed file size of the ROIs of the large and small frames (the ROIs in both large and small frames are exactly same, but the size of the run length codes is different). Initially, the compressed file size of the ROI_small and ROI_large are almost similar (looks similar but actually there is a difference of 7 bytes due to run length codes, see the difference in the run length codes of small and large frames in Table 1) and then the difference increases with the increasing number of objects in the frames. The reason for this increase is the increase in the number of bytes for representing the run length codes of the large frames. For large frames the run length codes require higher number of bytes compared to the case of small frames. The crossing point between the curves ROI_small and Frames_small occurred for the case of 18 objects in the frames.

Figure 9 shows the compression efficiency of the Gzip for compressing ROIs detected in both large and small frames having varying number of objects (i.e. 0, 2, 4… 20 objects). The difference between compressed file size of the large frames and that of ROIs is high. The reason for this is that Gzip compression does not provide high compression ratio for large frames because it is dependent on the contents of the frames which is quite high in large frames.

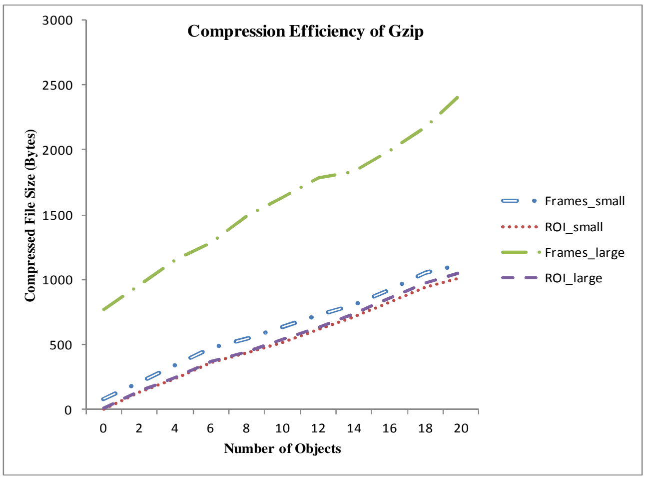

Figure 9. Performance of Gzip for ROI detection and coding method.

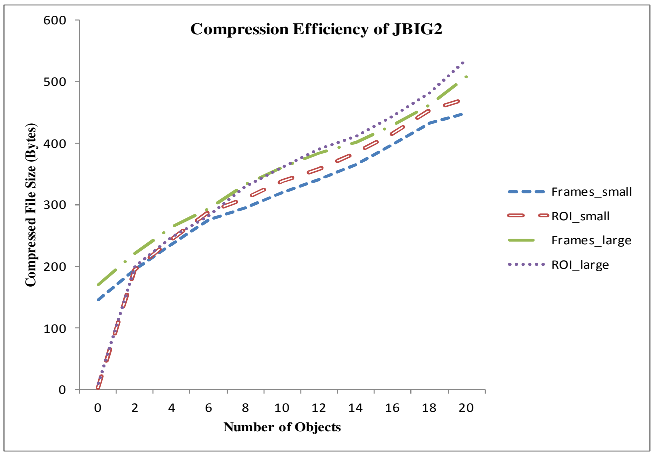

Figure 10. Performance of JBIG2 for ROI detection and coding method.

Figure 10 shows the compression efficiency of the JBIG2 for both large and small sized frames having different number of objects (i.e. 0, 2, 4 … 20 objects). Contrary to the results of Figure 8 and Figure 9, the difference between compressed file size of the large frames and that of ROIs is low. JBIG2 encodes the transitions from black to white pixels and vice versa which is exactly the same in both large and small frames. The curve representing ROI_small becomes greater than that of Frames_small for small frames having two or more objects on the average. The reason for this is the overhead involved due to run length codes.

Figure 10 shows that for small images JBIG2 is not giving any improvement for compressing ROIs. But instead the compressed file size of the ROIs is higher than that of the compressed file size of the small frames because of the extra run length codes needed for the reconstruction of the frames from the ROIs. It also shows that JBIG2 is offering little improvement for compressing ROIs in the large frames. If the number of objects in the large frames is more than 8 then JBIG2 is not giving any improvement for compressing the ROIs even in the large frames (Figure 10).

V. Discussion of the Results

It is clear from the results that the performance of ROI coding is high for large frames. In Section 4 we observed that the cross over point between the curves of ROI and frame for small frames occurred, but it did not occurred for large frames.

We are interested to find the crossing point between the curves of ROI and large frames also. For this purpose, we generated a new set of large frames with increasing number of large objects (objects with radius equal to 300). The reason behind increasing the size of objects in the frames is to increase the size of ROI_Image and ultimately achieve the crossing point between the curves of ROI and frames.

The ROI algorithm removes the rows and columns of the frames which are completely black. There are high number of complete black rows and columns in the frames having too few objects. Hence the part of the frames which exist in the detected ROI_Image is small for frames with few objects and is large for frames having more objects.

The new set of generated large frames is compressed with the three compression standards. The results are shown in Figure 11, where both horizontal and vertical axis is in log scale. In each part of Figure 11 the horizontal axis shows the percentage of the pixels remained in the detected ROI_Image from large frames (where 100 % means the whole frame is present in the ROI_Image).

The crossing point between the curves (average of 50 frames for each index of the horizontal axis) of the frames and ROIs (ROI_Image) occurred for the case where almost 73%, 93% and 100% pixels of the original frames remained in the ROIs for JBIG2, Group4 and Gzip respectively.

The Group 4 is a type of the generalized Tag Image File Format (TIFF) compression. Due to more versatile format of the TIFF header, it has many fields in its Image File Directory (IFD) and each field is 12 bytes long. The compressed file size for Group 4 for coding ROIs is high compared to JBIG2 because of the overhead involved due to many fields in its IFDs. We conclude that Group 4 is the suitable compression method for compressing ROIs because it offers high compression performance for both large and small frames and is computationally less complex.

We analysed the compression efficiency of the three compression methods for ROIs of nine different shapes of the objects and the results are shown in Figure 12. The compressed file size of the ROIs of different shapes is slightly different from each other. Also the compressed file size for compressing ROI of various shapes is different for the three compression methods.

Figure 12 shows that the different shapes of the objects have little effect on the performance of the ROI coding using JBIG2 compression standard (the compressed file size for frames with different shapes of the objects is almost similar).

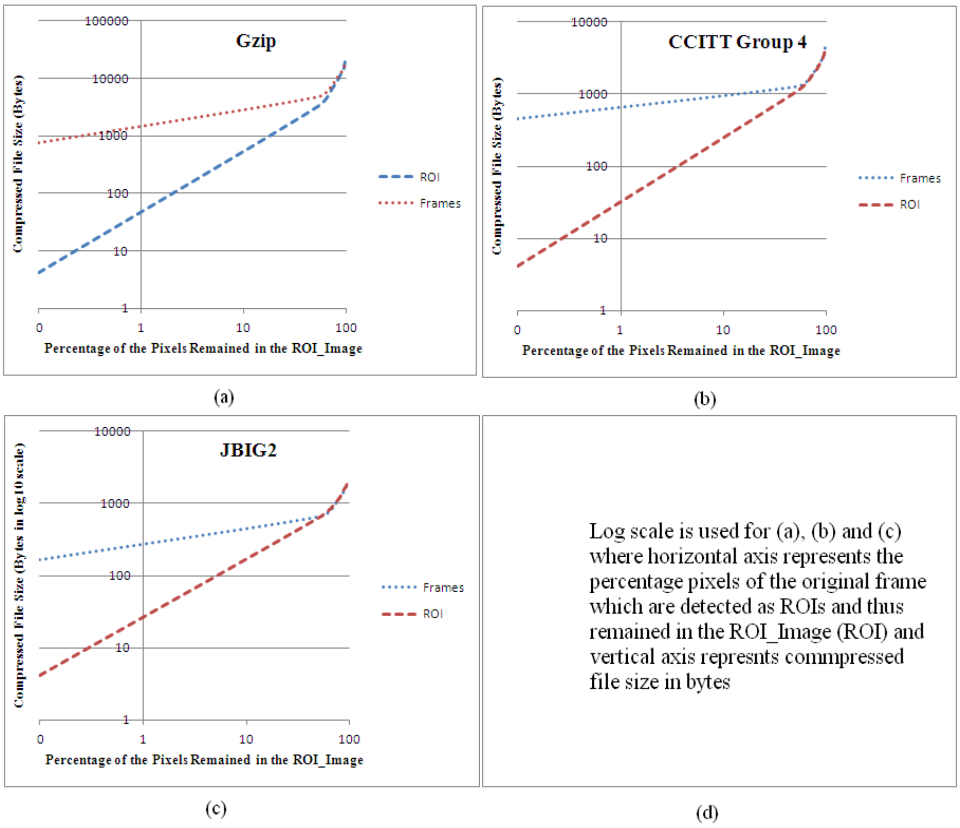

Figure 11. Crossing point between the curves of ROIs and large frames.

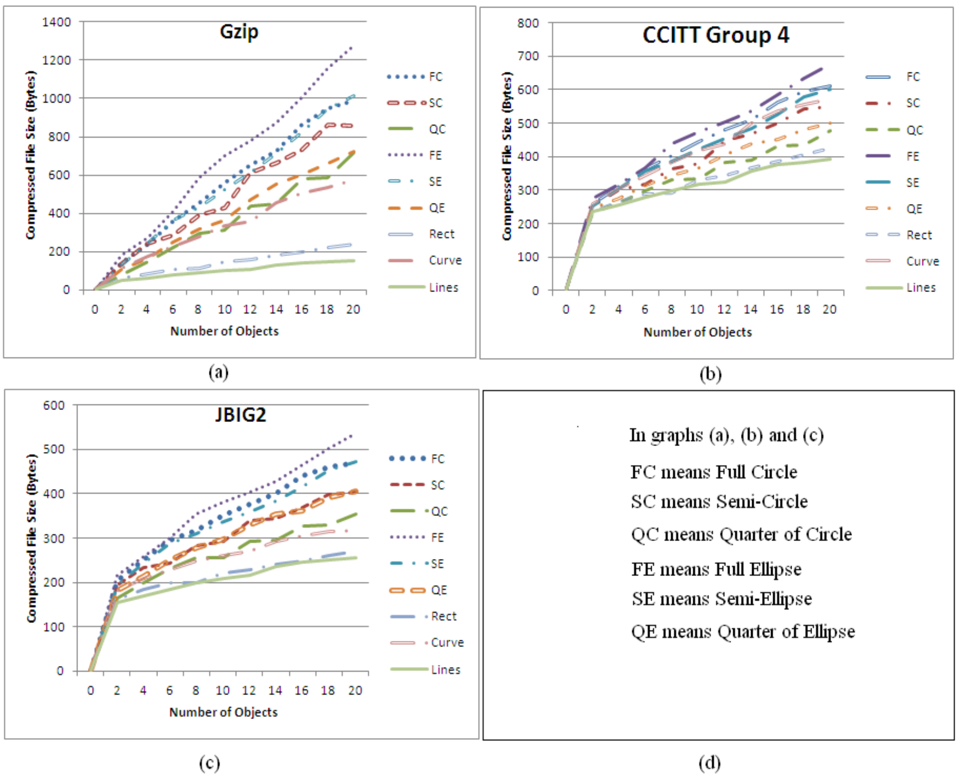

Figure 12. The compression performnace of ROI coding for various shapes of objects in the images.

Figure 12 shows that among the compression methods, Gzip is the most sensitive to the various shapes of the ROI in the frames. It also shows that compare to the other methods, the compressed file size of Gzip is the lowest for frames having very few objects (for one or two objects) and is the largest for frames having more than 16 objects. This behavior of Gzip also confirms that it is dependent on the contents of the frames (small compressed file size for few objects while large compressed file size for more objects).

Figure 12 (b) shows that the compressed file size for compressing ROIs of various shapes using CCITT Group 4 is almost similar. The most important fact that needs to be observed here is that compressing just one object (of any shape) using CCITT Group 4 results in quite large compressed file size. This confirms that the header information of CCITT group 4 is quite high, which must be investigated in future.

VI. CONCLUSION

In this paper, we analyzed the compression performance of ROI detection and coding methods for data reduction in WVSN. We considered the ROIs of different shapes and varying number of objects in large and small frames. We considered the three bi-level image compression methods including Gzip, JBIG2 and CCITT Group 4, based on their good compression efficiency. The results show that the effect of different object shapes on ROI coding is high for Gzip as compared to the JBIG2 and CCITT group 4. Moreover, the Gzip, using ROI coding, provides high compression efficiency compared to frame coding, for both large and small frames. But because of the underline compression algorithm, the compression performance of both CCITT group 4 and JBIG2 is better than that of Gzip for ROI coding. The JBIG2 offers good compression efficiency for coding ROIs in large frames but not in small frames. The CCITT Group 4 provides better compression performance for coding ROIs from both large and small frames. However, it provides larger sized compressed file than that of JBIG2 because of its large header information overhead. The CCITT Group 4 is preferable because it provides better compression efficiency for compressing ROIs from both large and small frames and it is computationally less complex than JBIG2. Thus, the ROI detection method in combination with CCITT Group 4 is a good strategy for data reduction in WVSNs. To achieve further data reduction, the study of the redundancy in the header information of the compression methods is suggested.

REFERENCES

[1] L. Ferrigno, S. Marano, V. Paciello and A. Pietrosanto, “Balancing computational and transmission power consumption in wireless image sensor networks”, IEEE Int’l Conf. on Virtual Environments, Human-Computer Interfaces, and Measurement Systems, Italy, July 2005.

[2] W., Chen, A., Hamid, K., Richard, “Real-Time Human Posture Reconstruction in Wireless Smart Camera Networks,” IPSN '08 Proceedings of the 7th international conference on Information processing in sensor networks 321-331. IEEE Computer Society Washington, DC, USA 2008, ISBN: 978-0-7695-3157-1.

[3] P. Kulkarni, D. Ganesan, P. Shenoy and Q. Lu, “SensEye : A multi-tier camera sensor network”, Int’l Multimedia Conference archive Proc. of the 13th annual ACM int’l conference on Multimedia Pages: 229 - 238 2005, ACM New York, NY, USA 2005, SBN:1-59593-044-2.

[4] K. Khursheed, M. Imran, M. O’Nils and N. Lawal, "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] K., Khursheed, M., Imran, A.W., Malik, M., O’Nils, N., Lawal, T., Benny, 2011. “Exploration of Tasks Partitioning Between Hardware Software and Locality for a Wireless Camera Based Vision Sensor Node,” Proceedings of the International Symposium on Parallel Computing in Electrical Engineering (PARELEC'2011).

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

[7] K. Khursheed, M. Imran, N. Ahmad, M. O’Nils, "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.

[8] 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.

[9] http://www.itu.int/itu-t/recommendations/index.aspx?ser=T

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

[11] D., Nister, C., Christopoulos, “Lossless region of interest coding,” (1999) Signal Processing, 78 (1), pp. 1-17, doi: 10.1016/S0165-1684(99)00044-4.

[12] A., Said, W.A., Pearlman, “A new, fast, and efficient image codec based on set partitioning in hierarchical trees,” (1996) IEEE Transactions on Circuits and Systems for Video Technology, 6 (3), pp. 243-250. doi: 10.1109/76.499834.

[13] A.P., Bradley, F.W.M., Stentiford, “JPEG 2000 and Region of Interest Coding,” (2002) Digital Image Computing Techniques and Applications (DICTA), Melbourne, Australia, pp. 303-308.

[14] R., Sundhu, “Intelligent Compression of Medical Images with Texture Information,” Computer and Automation Engineering (ICCAE), 978-1-4244-5587-7/10, 2010 IEEE.

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

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

[17] K. Khursheed, M. Imran, N. Ahmad, M. O’Nils, “Analysis of Change Coding for Data Reduction in Wireless Visual Sensor Network,” The paper is submitted for publication in IEEE Transactions on Industrial Informatics.

[18] K., Khursheed, N., Ahmad, M., Imran, M., O’Nils, “Performance-Constraints Analysis of Bi Level Image Compression Methods for Embedded Applications” Submitted to Journal of Real Time Image Processing.