An Image Compression Method for Fleck Board-3

Indrit Enesi[1], Betim Çiço[1]
[1]*Polytechnic University of Tirana, Albania*

***Abstract*-The recent availability of inexpensive hardware, such as CMOS and CCD cameras, has enabled the new research field of wireless multimedia sensor networks. This is a network of interconnected devices, capable of retrieving multimedia content from the environment. In this type of network, the nodes usually have very limited resources, in terms of processing power, bandwidth and energy. Efficient coding of the multimedia content is therefore important. A new image compression codec, suitable for this type of network was developed, a very high compression ratio can be achieved, the compression rate varies depending on the content but it is usually between 90% and 99% compared to the previous camera network.**

***Keywords*-wireless sensor network, Fleck, image compression, packet format, entropy coding**

1. Introduction

Wireless Sensor Networks (WSN), are a new tool to capture data from the natural or built environment at a scale previously not possible. Typical WSN have the capability for sensing, processing and wireless communication, all built into a tiny embedded device. The increasing availability of low-cost CMOS or CCD cameras and digital signal processors (DSP), means that more capable multimedia nodes are now starting to emerge in WSN applications. This has lead to recent research field of Wireless Multimedia Sensor Networks (WMSN) that will not only enhance existing WSN applications, but also enable several new application areas [1]. One of the applications areas for the WSN is smart farming, where a sensor network can be used for monitoring the environment, to make better use of the land and water resources, which is even the application area for this work.

2. Problem description

Systems have previously been developed to capture and send images back to a base station over a WSN, as shown in figure 1. The cameras are static and periodically capture an image that is transmitted back to the base station. The base station then inserts the images in a database that can be accessed over the Internet. The camera nodes, have limited resources in terms of processing power, bandwidth and energy. The energy consumption is a very important issue, since the nodes are battery powered and are supposed to run for a long time without the need of service from a user. To save power, the camera nodes, power down as much hardware as possible between images that are taken. It is also important to use efficient image compression algorithms, to reduce the time needed to transmit an image. This is also important, because the bandwidth for the wireless communication is only about 30-40 kbps between the nodes. The existing systems did not use any compression of images at the camera node.

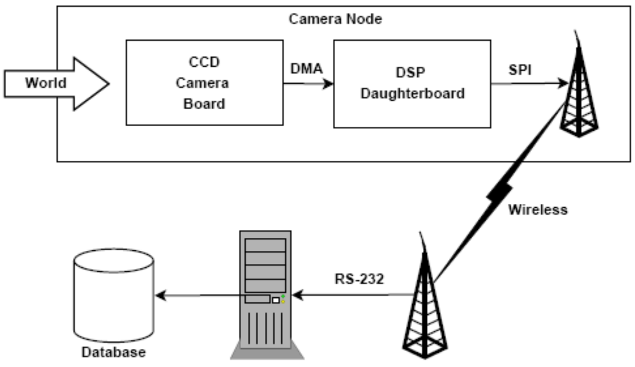


Figure 1. An overview of the camera wireless sensor network system

An uncompressed QVGA (320x240 pixels) image in the format used for transmission in the previous system, has a size of 153 600 bytes. This will take more than 40 seconds to transmit using 30 kbps radio. This is clearly a waste of energy and bandwidth, two very limited resources in the system. The aim of this paper, is to develop an efficient image compression scheme and integrate it into the current system. This project did not include any hardware work, since the existing FleckTM platform, could be used.

3. Related research

It is important for the agricultural industry to make optimal use of the resources. The modern farming is highly mechanized and involves very large areas per farmer. This makes it much more difficult for the modern farmer, to have good control of the land utilization.

Sensor networks have been developed to provide information about the farm. Recently there have been several studies in wireless multimedia sensor networks. Compared to other projects in WSN, this project focuses more on low-bit rate image transmission over long-range outdoor sensor networks. There are two general approaches for low-bit rate video coding. One is waveform-based coding and the other is model-based coding [2]. In the model-based coding technique, a model of the three-dimensional scene is created, and then the 2-D images are analyzed and synthesized. This technique can achieve very low bit rate. It is however a very complicated technique and for this reason it is not interesting for this paper. Instead the waveform-based approach is used where the compression is done on the 2-D image plane without considering the underlying 3-D scene. This approach is also used by all image and video coding standards today.

The most commonly used video encoding standards today, such as MPEG-4 [3] and H.264 [4], are based on the block-based hybrid coding. Here the encoder has much higher complexity compared to the decoder, usually 5-10 times higher [5]. For sensor networks the reversed relation is desired since the encoder usually runs on a battery powered device having very limited resources, whereas the decoder usually runs on a powerful computer. It is known from theoretical bounds established by Sleipian and Wolf for lossless coding [6] and by Wyner and Ziv [7] for lossy coding that efficient encoding can be achieved with the reverse complexity between encoder and decoder.

These techniques are usually referred to as distributed source coding. Recently there has been research to make use of this coding for sensor networks [8]. Although this is an interesting approach for image coding in sensor networks it is still much research left to do in this area before it really becomes useful in practical implementations. For this reason the distributed source coding approach was not considered.

4. Hardware Platform

4.1 Fleck3

The wireless sensor network hardware platform used in this project was the FleckTM[9].For software developing the TinyOS[10] operating system can be used on this platform. FleckTM has been especially designed for use in outdoor sensor networks for environmental monitoring and for applications in agriculture. Some of the features that make this platform suitable for outdoor deployments are long-range radio and integrated solar-charging circuitry.

For wireless sensors network the energy is a very important topic. To make use of all energy from the energy source a DC-DC converter is used. This enabled the FleckTM to work down to very low supply voltage. On the supply side a good source of energy are solar panels, especially in a sunny environment like Albania. The FleckTM has integrated solar-charging circuitry making it possible to connect a solar cell directly to one pair of terminals and the rechargeable battery to another pair. The latest version, Fleck-3, is based on the Fleck-1c. The main improvement in Fleck-3 is a new radio, NRF905, capable of operating in all ISM bands (433MHz, 868MHz and 915MHz). This radio transceiver takes care of all framing for both sent and received packets, so the load on the microcontroller is therefore reduced. It also has a more sensitive radio compared to the Fleck-1c enabling a range of about 1km. The Fleck-3 platform is also designed for easy expansion via add-on daughterboards which communicate via digital I/O pins and the SPI bus.
The two key daughterboards used in this project are described below.

4.2 DSP Board

To be able to handle high-bandwidth data, like audio and video, a board having a second processor was designed. Communications with the Fleck-3 baseboard is attained by SPI bus. Here the DSP board is the slave and the FleckTM is the master. This works fine if the processing time on the DSP is fixed, for instance when capturing and transferring an uncompressed image from the camera. In this configuration there is no way for the DSP board to inform the FleckTM when it has finished compressing an image. Instead the FleckTM must poll the DSP board to get information about the processing of an image is completed. Communication with the camera is controlled by a FPGA. The DSP board sets parameters on the camera using the IIC bus. The transmission of an image from the camera to the SRAM on the DSP board is done by DMA.
The smallest size data accessible in memory on this DSP is 16-bit but the pixel values from the camera are sampled in 8-bit. To avoid using shifting and masking when accessing the stored pixel values, each 8-bit value is stored as a 16-bit value. This means that the usable memory is basically reduced to half that is 512kB.

4.3 Camera Board

The camera board has an Omnivision VGA 640x480 colour CCD sensor (OV7640). The sensor is progressive scan and supports windowed and sub-sampled images. The camera parameters can be set by the DSP over the local IIC bus. An FPGA on the DSP board is used to move the image into external SRAM.

Combining a FleckTM with the DSP board and a camera board creates a highly functional network camera node. To save power, the DSP and the camera have to be turned off most of the time [12].When the camera is powered on, must be initiated using the IIC bus, this includes setting parameters for auto white balance and image format. The entire initiation process currently takes a few seconds, since the camera needs time to settle before the first image can be taken after initiation. This can be a. problem in applications where an image must be captured frequently. To take an image after the camera has been powered up and initiated the DSP toggles a GPIO pin. The FPGA will then get an image from the camera and DMA it into the SRAM.

5. Image Compression

5.1 Overview

The compression of an image at the sensor node includes several steps. The image is first captured from the camera. It is then transformed into a format suitable for image compression. Each component of the image is then split into 8x8 blocks and each block is compared to the corresponding block in the previous captured image, which iscalled the reference image. The next step of encoding a block involves transformation of the block into the frequency plane. This is done by using a forward discrete cosine transform (FDCT). The reason for using this transform is to exploit spatial correlation between pixels. After the transform most of information is concentrated to a few low-frequency components. To reduce the number of bits needed to represent the image, these components are then quantized. This step will lower the quality of the image by reducing the precision of the components. The trade off between quality and produced bits can be controlled by the quantization matrix, which will define the step size for each of the frequency component. The components will also be ZigZag scanned to put the most likely non-zero components first and the most likely zero components last in the bit-stream. The next step is entropy coding. We use a combination of variable length coding (VLC) and Huffman encoding. Finally data packets are created suitable for transmission over the wireless sensor network. And overview of the encoding process is shown in Figure 2.

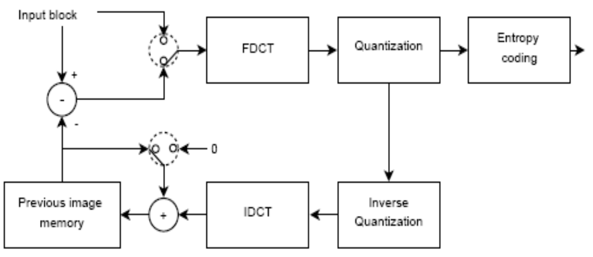


Figure 2. Overview of the encoding process for an image block

5.2 Pre-processing

Processing of the image is done on blocks of 8x8 pixels. To generate these blocks the image is first divided into macro-blocks having 16x16 pixels. The first macro-block is located in the upper left corner and the following blocks are ordered from left to right and each row of macro-blocks is ordered from top to bottom. Each macro-block is then further divided into blocks of 8x8 pixels. For each macro-block of a YUV 4:2:0 image there will be four blocks for the Y component and one block for each of the U and V component. The order of the blocks within a macro-block is shown in Figure 3. A QVGA image in the format YUV 4:2:0 will have 20 macro-blocks in horizontal direction and 15 macro-blocks in vertical direction, a total of 300 macro-blocks. Each of the macro-block contains six blocks. The image will therefore have a total of 1 800 blocks.

5.3 Prediction

Since the camera is static, in this application it is possible to make use of the correlation between pixels in the temporal domain, between the current image and the reference image. The previous encoded and decoded frame is stored and used as reference frame.

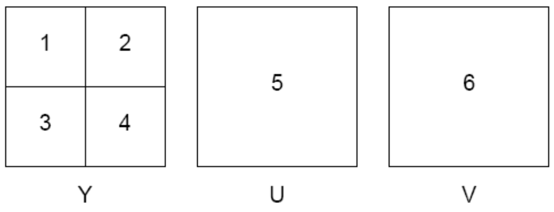


Figure 3. The order of blocks within a macro-block

Three approaches are taken in the way blocks are dealt with at the encoder (camera node) side: Skip block: If this block and the corresponding reference block are very similar then the block is not encoded. Instead only information that the block has been skipped and should be copied from the reference

image at the decoder side is transmitted. Intra-block encode: Encoding is performed by encoding the block without using any reference. Inter-block encode: Encoding is performed by encoding the difference between the current block and the corresponding block in the reference frame.
First the mean-square error (MSE) between this block and corresponding block in the reference frame is calculated.
If the current block is denoted by 1 and the reference block by 2 the MSE is given by:

$$MSE = \frac{1}{N}\sum_{m,n}\left(\psi_1(m,n) - \psi_2(m,n)\right)^2 \quad (1)$$

where N is the total number of pixels in either block. If the MSE is below some threshold the block is encoded as a skip-block and no further processing of the block is performed. For non-skip blocks both intra and inter-block encoding is performed where the type producing the least bits is selected for transmission. For the inter-block encoding a new block containing the difference between the current block and the corresponding reference block is first created. For the difference block the notation 3 is used. This block is given by:

$$\forall_{m,n,}\psi_3(m,n) = \psi_1(m,n) - \psi_2(m,n) \quad (2)$$

For inter-block encoding, this is the block used for further processing. For the intra-block encoding the current block is used for further processing without any prediction to reference blocks. A header is added to each block that will inform the decoder about the type of block. The header for a skip block is one bit and for non-skip blocks two bits. Since both the encoder and the decoder must use the same reference for prediction. The encoder must also include a decoder. In the case of a skip-block the reference block is not updated. For both intra-encoded and inter-encoded blocks, the block is decoded after encoding and the reference image is updated.

5.4 *Discrete Cosine Transform*

In this step each block is transformed from the spatial domain to the frequency domain. This will generate a matrix of 8x8 frequency coefficients. After this transformation, most information in the block will be concentrated to a few low-frequency components. The forward DCT transform is given in Eq.(3) and the inverse DCT transform is given in Eq.(4).

$$F(u,v) = \frac{1}{4}\Lambda(u)\Lambda(v)\sum_{x=0}^{7}\sum_{y=0}^{7}\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}f(x,y) \quad (3)$$

$$f(x,y) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7}\Lambda(u)\Lambda(v)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}F(u,v) \quad (4)$$

where:

$$\Lambda(\xi) = \begin{Bmatrix} \frac{1}{\sqrt{2}}, \mathit{for} \rightarrow \xi = 0 \\ 1, \rightarrow \mathit{otherwise} \end{Bmatrix}$$

Since energy consumption is of high priority on battery powered sensor nodes, an efficient algorithm for the DCT transform is desired. This is an important problem in the video and image coding community, therefore fast DCT algorithms have been studied extensively. In this project a DCT transform presented in a paper by Loeffler,[11] was used. To make it run efficiently on a DSP, the implementation was done by only using integer multiplications. To do a DCT transform of a block the total number of 8-point transforms used was 16.

5.5 *Quantization*

The information for the chrominance coefficients is also reduced more compared to the luminance coefficients. The matrices for the step-size used for different coefficients are shown in Table 1. The quantizer step-size for each frequency coefficient, S_{uv}, is the corresponding value in Q_{uv}. The uniform quantization is given in:

$$Sq_{uv} = round\left(\frac{S_{uv}}{Q_{uv}}\right) \quad (5)$$

where$Sq_{uv}$is the quantized frequency coefficient. To enable the efficient entropy coding of the quantized frequency coefficients, the order of which they are stored is changed.

Table 1. Quantization matrix for the luminance and chrominance coefficients.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

The purpose of this is to move the coefficients that are more likely to be zero towards the end, known as zigzag scan.

5.6 Entropy Coding

To reduce the number of bits needed to represent the quantized coefficients, the code words are encoded using a combination of run-length encoding (RLE) and Huffman encoding. For the DC coefficient, it is first predicted from the DC coefficient in the previous block. To make the encoding less sensitive for losses, the DC coefficient in the first block in each row of blocks is not predicted. The coefficient is divided into two parts. One part is a symbol for the category and the other part is additional bits to describe the value within a category. The symbol for the category is encoded using a Huffman codeword. The AC coefficients are encoded using a combination of RLE and Huffman encoding. The quantized coefficients usually contain many zero values. To encode these efficiently each Huffman symbol has two parts. The first part is the run-length. This is the number of zero coefficients in front of the first non-zero coefficient. The next part is the category for the non-zero coefficient. These two parts are labelled and sized respectively in the Huffman table. The symbol is encoded using Huffman a codeword. This is followed by a number of bits to describe the value of the coefficient within a category. The maximum run-length is 15 and if the number of consecutive zero coefficients is more than this number, a special codeword is used. This is the ZRL code and it represents 16 consecutive zero coefficients. If all remaining coefficients have the value zero a special codeword is used. This is the end of block (EOB) codeword.

5.7 Huffman table

Currently a fixed and pre-shared Huffman table is used for encoding and decoding the DC and AC coefficients. Different tables are used for AC and DC components and different tables are also used for luminance and chrominance coefficients, a total of four tables. These are the typical Huffman tables given in the Jpeg [13] standard and they are calculated based on the statistical properties from a large amount of images. To prepare for the feature of sending custom Huffman tables from the encoder to the decoder, the tables are compressed at the encoder side. A compressed table contains two lists. One list is for the number of code words of each length and the other list is a list of symbols. When the Huffman table is decoded a binary tree is built. To assign code words to symbols, the tree is traversed using depth first. When a vertex is visited, it will be assigned a symbol, if all symbols of the given level (code length) have not already been assigned. A vertex that has an assigned symbol will become a leaf in the tree. The code for each symbol, is built up from the path, beginning from the root of the tree to the vertex. The first path visited from a vertex will be assigned the bit “0” and the second path visited will be assigned the bit “1”.

The uncompressed Huffman tables are stored as a list of code words, containing both the actual codeword and the length of the codeword. For encoding this is a good data structure since the encoding process involves assigning a codeword to a symbol and this can be directly looked up in the list. For decoding a tree would be a much better data structure. However since the decoding is done on a node where computational resources and energy consumption is not an issue, a list is also used at the decoder side.

6. Wireless communication

A protocol was developed to transport compressed images from the camera node back to the base station. This protocol must be able to fragment the image to small packets on the sender side and reassemble the packets at the receiver side. The FleckTM nodes used for wireless communication have a maximum packet size of 27 bytes. The Huffman encoding used in the image compression requires that the decoding starts from the beginning of a block. To make it possible, restart the decoding. If some data is lost in the transmission resynchronization, markers can be used. This will however add extra data and the encoding efficiency will be reduced. In this work no resynchronization markers are added to the bit stream. Instead an integer number of blocks is put into each packet transmitted over the wireless channel. By using this method it is possible to decode each packet even if previous packets have been lost.

Table 2.Categories for the predicted value for DC coefficients

Symbol	Predicated value
0	0
1	-1,1
2	-3,-2,2,3
3	-7..-4,4...7
4	-15..-8,8..15
5	-31..-16,16..31
6	-63..-32,32..63
7	-127..-64,64..127
8	-255..-128,128..255
9	-511..-256,256..511
10	-1 023..-512,512..1023
11	-2 047..-1024,1 024..2 047

Table 3. Categories for the AC coefficients

Size	AC coefficients
1	-1,1
2	-3,-2,2,3
3	-7..-4,4..7
4	-15..-8,8..15
5	-31..-16,16..32
6	-63..-32,32..63
7	-127..-64,64..127
8	-255..-128,128..255
9	-511..-256,256..511
10	-1 023..-512,512..1 023

Packet format

Since packets can be lost each packet must contain information about which part of the image data that is included in the packet. For this an offset to the first block in the packet is stored. This is the offset from the first block in the images. The number of blocks in the packet is also stored in the packet header. Since the base station should be able to handle images from several camera nodes, a field informing the receiver about the sending node is needed. Each node can have several sensor sending data. For instance one node can have both a camera and a temperature sensor. For this reason there must also be a field informing the receiver about the type of data, that is included in the packet.
The total packet header is 8 bytes. Since the maximum packet size is 27 bytes, this is a relatively high packet overhead.

7. Error Control

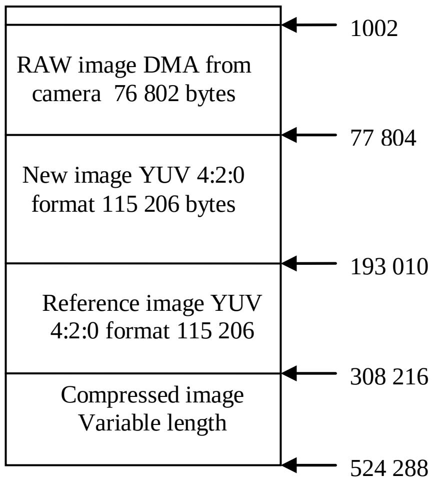
Another error control method used in this project at the encoder side is to periodically force each block to be encoded as an intra block. Because of the predicatively coding technique used, a lost block will not only cause error in the current image, but the error will also continue to propagate into the following frames. By sending an intra block, that is not predicatively coded, the error propagation will stop. The maximum number of non-intra blocks to be encoded before a block is forced to intra mode, can be set. The maximum amount of forced intra blocks per frame can also be set, to avoid the size of the encoded images varies too much.

8. Implementation

A system that sends uncompressed images from a camera node to a base station had been developed. The existing source code for the DSP to grab an image could be reused. On the DSP the major part was to implement the compression of an image. For wireless communication the protocol to schedule cameras and other functionalities, could also be reused. The packet format had to be changed to handle the variable length data that followed from the compression of images. This was also true for the communication between the DSP daughterboard and the FleckTM node. At the base station the decoding of images, has to be added. The encoder was implemented in C on the DSP and the decoder was implemented in JAVA at the base station. To simplify the development process, the encoder and decoder was first implemented in a Linux program using C. This made testing of new algorithms and bug fixing much easier. For testing pre-stored images from the sensor network was used. After the Linux version was developed, the encoder was moved to the DSP and the decoder was moved to the Java application.

9.1 DSP memory usage

The image is first transmitted from the camera to the SRAM on the DSP using DMA. A new image is then created using a more suitable format for image processing. This is the YUV 4:2:0 format, where chrominance and luminance information is separated. Each pixel value is stored using 16-bit values to avoid shifting and masking, each time encoder on the DSP data is accessed. Since prediction is used for image encoding a reference image must also be stored. This image has the same format as the previous image. Finally the compressed image is also stored in the SRAM on the DSP daughterboard. The memory map is shown in Figure 8. The communication between the FleckTM and the DSP is performed over a SPI bus. The protocol is based on a one byte command sent from the FleckTM to the DSP. Each command can have a number of bytes for parameters and for return values from the DSP to the FleckTM.



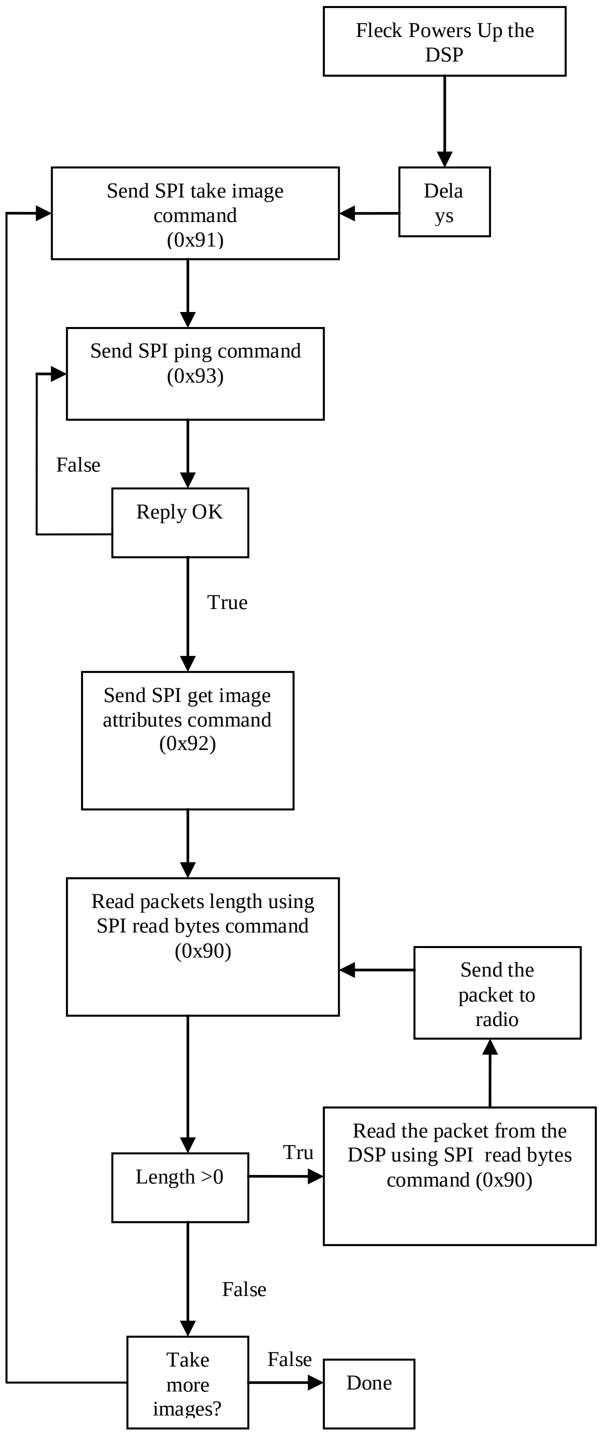
1002
RAW image DMA from camera 76 802 bytes
77 804
New image YUV 4:2:0 format 115 206 bytes
193 010
Reference image YUV 4:2:0 format 115 206
308 216
Compressed image Variable length
524 288

Figure 4.The memory map for the image encoder on the DSP

Several different SPI commands were defined for taking an image and for transferring the data from the DSP to the FleckTM.

9.2 Taking images

The image taking process is initiated from the FleckTM. It first powers up the DSP and waits for a while to give the DSP time to start up. After this a command is sent to ask the DSP to take an image. The DSP will now send a command to the camera to capture an image and transmit it to the DSP using DMA. After this the DSP will compress the image and store the compressed image in the SRAM.
Since the time to capture and compress the image can vary, the FleckTM must poll the DSP to ask it if the encoding of the image is completed. Once the compressed image is available on the DSP, the FleckTM starts to transfer the image from the DSP to the FleckTM over SPI.
The compressed image on the DSP is divided into packets, suitable for transmission over the wireless channel. The first byte is the length of the packet followed by the packet data. A length of zero is used to terminate the sequence. The flow chart for taking images, is shown in Figure 5.



Fleck Powers Up the DSP
Dela ys
Send SPI take image command (0x91)
Send SPI ping command (0x93)
False
Reply OK
True
Send SPI get image attributes command (0x92)
Read packets length using SPI read bytes command (0x90)
Send the packet to radio
Read the packet from the DSP using SPI read bytes command (0x90)
Tru
Length >0
False
Take more images?
False
Done

Figure 5. Flow chart for taking images

9. Results

The result of this work, was a system to transmit compressed images from a camera node over a wireless connection to a database, that could be accessed using the Internet. The compression rate varies, depending on the content but is usually between 90% and 99%, compared to the previous camera network. The system can handle several cameras and when more cameras are added, they will be scheduled to take and transmit images at different times. Since it is expensive, in terms of energy consumption to power up and power down the camera, it will take a sequence of images each time it is powered up. These images are then reconstructed to a video clip at the base station. The number of images to take each time the camera powers up can be set as a parameter, when the system is started. The first image in the sequence and the video clip is added to the database to be accessible over the Internet. Since power saving features was not used here and the cameras were always powered on, it was possible to use very tight scheduling between cameras. The reason for this modification was to create a more interesting demonstration, compared to the real system, where nodes are put to sleep most of the time to save power.

10. Future works

This work had a focus on the image coding for sensor networks. In the future work, it can be interesting to look more at the wireless communication in an image sensor network. The error control for images and video in wireless network usually requires a cross-layer design between the coding and the network. The overhead from the packet header is relatively large since the maximum packet size used in the system, is small. Methods to reduce the packet header overhead are a possible future research topic. Another topic is on transport and routing layer, when the burst video or image data is mixed with the other sensor data. For future long-term research in this area, distributed source coding certainly is interesting.

References

[1] **Akyildiz I. F., Melodia T.,Chowdhury K., 2007**: A survey on wireless multimedia sensor networks. Comput. Networks, 51(4):921–960, 2007.

[2] **Lundmark H. Li, A.,Forchheimer R., 1994:** Image sequence coding at very low bit rates: a review. IEEE Transactions on Image Processing, 3(5):589–609, 1994.

[3] **ISO/IEC 2001:** 14496-2 information technology-coding of audio-visual objects. International Organization for Standardization, December 2001, Second edition.

[4] **ISO/IEC2005:**14496-10:2005 information technology coding of audio-visual objects part 10: Advanced video coding. International Organization for Standardization, 2005.

[5] **Girod B.,Aaron A., Rane Sh., Rebollo-Monedero D., 2005:**Distributed video coding. Proceedings of the IEEE, Volume: 93, Issue: 1:71– 83, January 2005.

[6] **SlepianD., Wolf J. K., 1973:** Noiseless coding of correlated information sources. IEEE Transactions on Information Theory, Volume: 19, Issue: 4:471– 480, July 1973.

[7] **WynerA. D.,Ziv J., 1976:**The rate-distortion function for source coding with side information at the decoder. IEEE Transactions on Information Theory, Volume: 22, Issue: 1:1– 10, January 1976.

[8] **Puri R., Majumdar A., Ishwar P.,Ramchandran K., 2006:** Distributed video coding in wireless sensor networks. Signal Processing Magazine, IEEE, 23(4):94–106, July 2006.

[9] **Wark T., Corke P., Sikka P., Klingbeil L., Guo Y., Crossman Ch., Valencia P., Swain D., Bishop-Hurley G.., 2007:** Transforming agriculture through pervasive wireless sensor networks. IEEE Pervasive Computing, 6(2):50– 57, 2007.

[10] **Hill J., Szewczyk R., Woo A., Hollar S., Culler D., Pister K.., 2000:** System architecture directions for networked sensors. SIGPLAN Not., 35(11):93–104, 2000.

[11] **Loeffler Ch., Lieenberg A., Moschytz G. S., 1989:** Practical fast 1-d dct algorithms with 11 multiplications. In International Conference on Acoustics, Speech, and Signal Processing (ICASSP-89), volume 2, pages 988–991, Glasgow, UK, 23-26 May 1989.

[12] **Polastre J., Szewczyk R., Culler D., Telos 2005:** enabling ultra-low power wireless research. In IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks, page 48, Piscataway, NJ, USA, 2005. IEEE Press.

[13] **Information technology**, **1992**: digital compression and coding of continuous-tone still images requirements and guidelines. ITU Recommendation T.81, 09 1999.

Corresponding author: Betim Çiço.
Institution: Polytechnic University of Tirana, Albania.