Poster Abstract: Practical issues in image acquisition and transmission over wireless sensor network

F.J. Molina, J. Babancho, J.M. Mora, C. Leon
University of Seville
Department of Electronic Technology
C/ Virgen de África, 7. Seville 41011 (Spain)
fjmolina@us.es, jbarbancho@us.es, jmmora@dte.us.es and cleon@us.es

ABSTRACT

Multimedia data have become an important objective in wireless sensor networks. Due to the node resource constraints (energy consumption, memory capacity, network latency and throughput) the incorporation of image sensor at the nodes is currently a challenge.

In this paper, we study different node architectures, evaluating processing time, energy consumption, image quality and data delivery issues. The study shows that a specialized image co-processor is an optimal solution[1].

1. ARCHITECTURES AND IMAGE COMPRESSION

In Wireless Sensor Networks (WSNs) a node can contain several sensors (temperature, humidity, solar radiation, accelerometer, etc). This makes this technology suitable for many applications where the nodes can be attached (e.g. environmental and body monitoring, industrial automation or home automation).

In general, image delivery over a WSN involves five steps: image acquisition, storing, processing, compressing and delivery. Resource constraints force all these steps to be optimized. The main optimization objectives are 3: to maximize the quality of the transmitted image and the network lifetime, and minimize the transmission delay. The optimization of the network protocol is being widely studied in the literature [1], [2]. However, image acquisition, storing, processing and compressing are becoming more and more important and should be studied in depth.

In this paper, we analyze two different hardware architectures to obtain a preliminary comparative analysis about resources optimization and system performances. Specially at the source node because requires higher performances. The first one is shown at figure 1: a single node connected to a CMOS image sensor. Most of these sensors allow a command-based dialog by a serial interface (I2C or SPI). Typically, it is possible to select the image format (CIF, GCIF), the data format (e.g. YCbCr 4:2:2, GRB 4:2:2 or RGB Raw Data), camera parameters (exposure, gain and white balance) and image quality parameters (brightness, contrast, saturation, sharpness gamma, and windowing). Any other operation or algorithm must be executed on the node microcontroller, and the original and processed images have to be saved to the internal memory.

Memory requirement is the first problem to analyze. One of the lowest standard image sizes, the QCIF (176 x 144) needs more than 64K for RGB color images, but most motes platforms have less than 32K of RAM memory. A row image can be delivered without processing if it is split *on the fly* (i.e transmitting each row after being read from the camera). Just one row should be saved in memory either in the source node or in any other intermediate node. But the energy requiered to deliver the image through the network is high, roughly to the image size (N^2). Furthermore, a packet loss can degrade the image at the sink node, though a simple interpolation algorithm can be used to estimate the lost pixels and reconstruct the image. The reliability of this scheme can be increased if the rows are also splitted into two packets with the odd and even pixels, respectively. Choosing different paths for both increases the probability of a packet loss but also one of them will be more likely to reach the sink node. Again, lost pixels can be estimated using an interpolation algorithm,

[1]This work has been supported by the Conserjería de Innovación, Ciencia y Empresa, Junta de Andalucía, Spain, through the excellence project ARTICA (reference number P07-TIC-02476).

but in this case, the real pixels are uniformly distributed throughout the image, so the quality of the image reconstructed by interpolation increases.

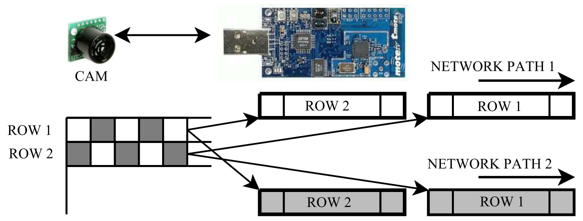

Figure 1: Reference architecture and an image splitting scheme

A compression algorithm can reduce this energy significantly , but no *splitting on the fly scheme* is possible, because the original and compressed images must be saved to memory. As we propose in the first architecture, an external memory (i.e an I2C RAM) can solve the problem. The new chip increases the energy consumption in three ways: the chip power, the time needed to transfer data from the camera (seconds), and a factor caused by the inefficient data access during the compression algorithm. All of them are insignificant compared to the required processing time. To test it, we have implemented a Haar transform for a grey QCIF image in NesC for TinyOS. It takes 80 seconds!

The energy difference of this solution to the *"on the fly scheme"* can be approximated to:

$$\Delta E \approx N^2 \cdot E_{BB}\,(1 - R_C) + T_P \cdot PWC$$

where R_C is the compression rate, E_{BB} the energy to broadcast a byte, T_P the processing time, and PWC the CPU power consumption for the processing time.

The first factor, R_C, is directly related to the image quality. The Haar transform [3] reduces the image size with a reasonable computation cost and high quality. We measure this quality with a classical Peak Signal-to-Noise Ratio (PSNR). For usual values of the PSNR (25-40dB), the figures show that the compression ratio can reach up to 98%. In addition to consumption efficiency, the memory required to host the compressed image matches with that of most of the node platforms, so that no extra memory is needed in the intermediate nodes.

To reduce the processing time, we proposed a different architecture: a high performance microcontroller acting as *image co-processor*. Even though the power consumption of this co-processor is much more higher, the processing time is substantially lower (the same experiment with the Haar transform takes just 2 seconds!)

The microcontroller selected was the PIC32XMX-460

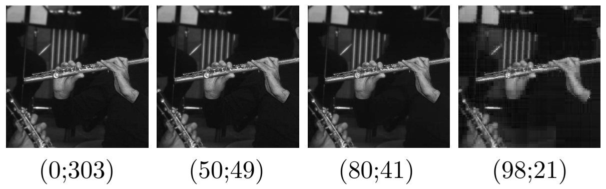

Figure 2: Original and compressed images: (CR; PSNR(dB))

with 32K RAM. In this case, the camera can be connected directly to the co-processor, so no external memory is needed, and in consequence, no delay time to transfer the pixels, no extra time to data access, etc.

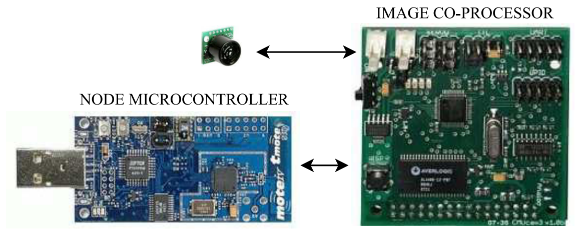

Figure 3: Co-processor architecture

In a future study, we will analyze an FPGA to implement the image co-processor. The capacity of a FPGA to execute parallel simple operations matches with the spacial redundancy of many image processing algorithms. Optimizing these algorithms for FPGAs, we expect to reduce the processing time to milliseconds. Compared with ASICs, the power consumption of an FPGA is considerably higher. However, an FPGA can be reprogrammed to update it to a specific application.

2. REFERENCES

[1] Y. Guo, Q. Wang, H. Huang, W. Tan, and G. Zhang, "The research and design of routing protocols of wireless sensor network in coal mine data acquisition," in *International Conference on Information Acquisition. ICIA'07*. IEEE, July 2007, pp. 25–28.

[2] J. Fonda, M. Zawodniok, S. Jagannathanand, and S. Watkins, "Development and implementation of optimized energy-delay sub-network routing protocol for wireless sensor networks," in *IEEE International Symposium on Intelligent Control*. IEEE, October 2006, pp. 119–124.

[3] P. Raviraj and M. Sanavullah, "The modified 2D-Haar transformation in image compression," *Middle-East journal of Scientific Research*, vol. 2, no. 2, pp. 73–78, 2007.