2011

Energy-efficient Foreground Object Detection on Embedded Smart Cameras by Hardware-level Operations

Mauricio Casares
University of Nebraska-Lincoln, mauricio.casares@huskers.unl.edu

Paolo Santinelli
University of Modena, paolo.santinelli@unimore.it

Senem Velipasalar
University of Nebraska-Lincoln, velipasa@engr.unl.edu

Andrea Prati
University of Modena, andrea.prati@unimore.it

Rita Cucchiara
University of Modena and Reggio Emilia

IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2011
doi: 10.1109/CVPRW.2011.5981838

Energy-efficient Foreground Object Detection on Embedded Smart Cameras by Hardware-level Operations

Mauricio Casares[1], Paolo Santinelli[2], Senem Velipasalar[1], Andrea Prati[2] and Rita Cucchiara[2]
[1]University of Nebraska-Lincoln, Dept. of Electrical Engineering
mauricio.casares@huskers.unl.edu, velipasa@engr.unl.edu
[2]University of Modena and Reggio Emilia, Modena Italy
paolo.santinelli@unimore.it, andrea.prati@unimore.it

Abstract

Embedded smart cameras have limited processing power, memory and energy. In this paper, we introduce two methodologies to increase the energy-efficiency and the battery-life of an embedded smart camera by hardware-level operations when performing foreground object detection. We use the CITRIC platform as our embedded smart camera. We first perform down-sampling at hardware level on the micro-controller of the image sensor rather than performing software-level down-sampling at the main microprocessor of the camera board. In addition, we crop an image frame at hardware level by using the HREF and VSYNC signals at the micro-controller of the image sensor to perform foreground object detection only in the cropped search region instead of the whole image. Thus, the amount of data that is moved from the image sensor to the main memory at each frame, is greatly reduced. Thanks to reduced data transfer, better use of the memory resources and not occupying the main microprocessor with image down-sampling and cropping tasks, we obtain significant savings in energy consumption and battery-life. Experimental results show that hardware-level down-sampling and cropping, and performing detection in cropped regions provide 54.14% *decrease in energy consumption, and* 121.25% *increase in battery-life compared to performing software-level down-sampling and processing whole frames.*

1. Introduction

Wireless embedded smart cameras are stand-alone units that combine sensing, processing and communication on a single embedded platform. Rather than transferring all the data to a back-end server, they can process images, and extract data locally. Even though battery-powered embedded smart cameras provide a lot of flexibility in terms of camera quantities and placement, they have limited resources, such as computational power, memory and energy. Since battery-life is limited, and video processing tasks consume considerable amount of energy, it is essential to have lightweight algorithms, and methodologies to increase the energy-efficiency of each camera. With the advances in hardware technology, embedded devices are becoming more sophisticated. Embedded smart cameras are being equipped with general purpose processing units that allow implementing sophisticated vision algorithms on these platforms.

Fleck et al. [4] present a network of smart cameras for tracking people. They use commercial IP-based cameras, which consist of a CCD image sensor, a Xilinx FPGA and a Motorola PowerPC. Cameras communicate via Ethernet connections. Quaritsch et al. [7] employ smart cameras with multiple DSP processors for data processing. Bramberger et al. [1] presented a smart camera architecture reaching a processing power of 9600 MIPS with onboard memory of 784 MB. While this high-end platform provides sufficient capabilities for image processing, it requires an average power consumption of 35 Watts.

Wired or IP-based cameras have powerful processing capabilities and relatively high bandwidth for communication. However, they have high power consumption and are larger in size. Many embedded vision platforms have been developed more recently. The Cyclops [9] and MeshEye [5] platforms have 7.3-MHz and 55-MHz processors, respectively. Thus, in both of these platforms the processing power is very limited. The camera mote introduced by Kleihorst et al. [6] has an 84MHz XETAL-II SIMD processor, and uses a higher resolution. The CMUcam2 [11] is a low-cost embedded camera with 75MHz RISC processor and 384KB SRAM. Due to the limited memory and processing power, only low-level image processing can be performed. Panoptes platform [3] hosts a 206-MHz processor, but has high energy consumption. SensEye [8] is a multi-tier network of heterogeneous wireless nodes and cameras. Rinner et al. [10] present a comparison of various smart camera platforms. Kerhet et al. [14] employ a hybrid architecture FPGA-MCU, where the processing load is distributed to increase the efficiency of the camera mote. Chen et al.[2] introduced the CITRIC camera mote that provides more computing power and tighter integration of physical components while still consuming relatively little power. An Omnivision sensor OV9655 is employed to capture frames. The down-sampling of the images is performed by software using the CITRIC API libraries.

Casares et al. [12] introduced an algorithm for resource-efficient foreground object detection, and presented the savings in processing time and energy consumption on CITRIC cameras. They obtained the savings in software-level. In this paper, our goal is to increase the energy-efficiency and the battery-life further by hardware-level operations when performing object detection. We present two methods to achieve this: (i) rather than performing down-sampling and image cropping at the main microprocessor on the camera board, we perform these operations at the micro-controller of the OV9655; (ii) we crop an image frame at hardware level by using the HREF and VSYNC signals at the micro-controller of the image sensor to perform foreground object detection only in the cropped search region instead of the whole image. Performing these functions at hardware level provides multiple advantages including savings in processing time and significant decrease in energy consumption. The amount of data, which is moved from the image sensor to the main memory at each frame, is greatly reduced. This, in turn, leads to significant savings in energy consumption thanks to the better use of the memory controller and the memory resources and not occupying the main microprocessor with performing image down-sampling and cropping at software-level.

We compare performing down-sampling and cropping by software using the CITRIC API libraries, and by hardware using the micro-controller of the image sensor. We present the experimental results showing the savings in processing time and energy consumption, and the gain in the battery-life when performing down-sampling and cropping operations at hardware-level, and processing only the cropped region of an image frame. Due to lack of compatibility between the device driver containing the existing kernel and the actual camera sensor, the results and savings are provided based on a single frame. Currently, a new kernel patch is under evaluation to capture successive cropped frames with adaptive window sizes.

2. The Embedded Smart Camera Platform

The wireless embedded smart camera platform employed in our experiments is a CITRIC mote [2]. It consists of a camera board and a wireless mote, and is shown in Figure 1. The camera board is composed of a CMOS image sensor, a microprocessor, external memories and other supporting circuits. The microprocessor $PXA270$ is a fixed-point processor with a maximum speed of 624 MHz and 256 KB of internal SRAM. The $PXA270$ is connected to a 64 MB of SDRAM and 16MB of NOR FLASH. Attached to the camera board is a TelosB mote from Crossbow Technology with a maximum data rate of 250 Kbps.

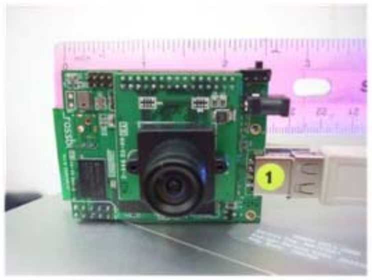

Figure 1. CITRIC camera: the wireless embedded smart camera platform employed in the experiments.

2.1. The Image Sensor

The image sensor on the CITRIC camera is a OmniVision OV9655 [13], which is a low voltage SXGA CMOS image sensor with an image micro-controller on board. It supports image sizes SXGA (1280×1024), VGA (640×480), CIF (352×288), and any size scaling down from CIF to 40×30, and provides 8-bit/10-bit data formats [2].

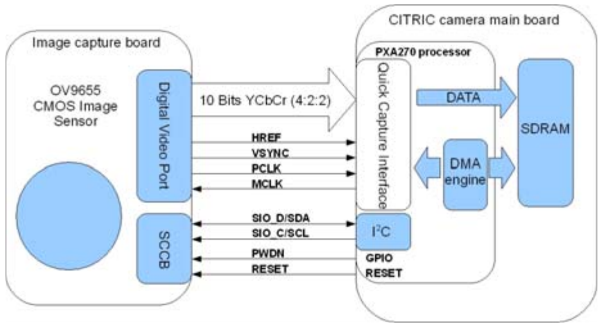

Figure 2. Interconnection of OV9655 and the Intel Quick Capture Interface on ARM PXA270.

The image sensor offers the full functionality of a camera and image micro-controller on a single chip. There is a complete control over image quality. Formatting, output data transfer and all required image processing functions are also programmable. The architecture and hardware interfaces of the OmniVision OV9655 are depicted in Fig. 2. The Serial Camera Control Bus (SCCB) interface is used to program the sensor behavior by setting all the control registers in the device. It is an Inter-Integrated Circuit (I2C) compatible hardware interface. The Digital Video Port provides a connection between the sensor and the CITRIC camera main processor PXA270. It is used to capture the image data. It is a unidirectional communication bus transferring

10-bit data signals and the line and frame synchronization signals [15].

2.2. The Quick Capture Interface

The main microprocessor PXA270 is an integrated system-on-a-chip microprocessor that incorporates a comprehensive set of system and peripheral functions. Some of these peripheral functions provide the ability to handle the image sensor. These are the Intel Quick Capture Interface, the DMA controller and the standard I2C interface. The quick capture interface provides a connection between the processor and the image sensor as seen in Fig. 2. It can acquire data and control signals, and performs the appropriate data formatting prior to routing the data to memory using direct memory access (DMA). The I2C interface is directly connected to the SCCB interface of the image sensor and is used to access the configuration registers set.

2.3. Frame Capture Operation

On the CITRIC camera platform, this interface operates in 10-bit Master Parallel mode. It requires a parallel data-bus interface, two control signals for frame timing and a pixel clock for basic timing. Master mode refers to a mode of operation in which the image sensor provides the line and frame synchronization signals. The line synchronization signal is commonly referred to as HREF or "line valid" and the frame synchronization signal is commonly referred to as VSYNC or "frame valid". For the Intel PXA270 master mode, this means that the line valid and frame valid signals are inputs to the quick capture interface. The sensor can be programmed for exposure, frame rate, and additional parameters. The programming is done through a separate interface, namely the I2C serial control interface. Once configured, the sensor begins providing data in addition to generating the frame and line synchronization signals. The MCLK signal output for the sensor is programmable. The timing signals VSYNC and HREF, provided by the sensor, activate and reset the quick capture interface that can be configured to provide an interrupt at the end of each line and each frame as shown in Fig. 3.

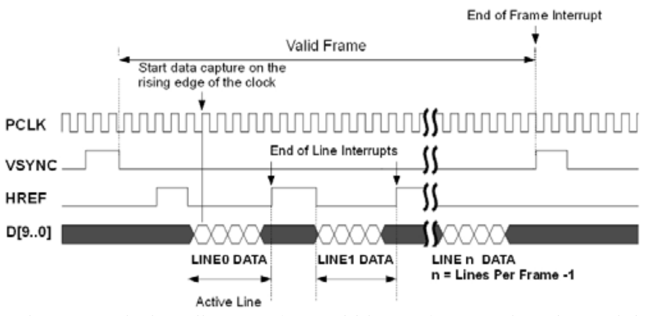

Figure 3. Timing diagram for grabbing a frame using the Quick Capture Interface.

3. Image Scaling and Cropping

As mentioned above, the main goal is to decrease the processing time and energy consumption. To achieve this goal, we perform two main operations at hardware level: (i) the change of the image resolution and (ii) image cropping based on a region of interest (ROI). The hardware subsystem composed of the image sensor and the quick capture interface is highly configurable. The exploitation of this flexibility by performing these functions at hardware level provides a reduction in the amount of data that is moved from the image sensor to the main memory at each frame. This, in turn, leads to significant savings in energy consumption thanks to the better use of the memory controller and the memory resources and freeing the main microprocessor from the tasks of performing image down-sampling and cropping at software-level. It is important to note that the achievement of this goal depends on the ability to implement the described functions in hardware. This has to be done by working on the image sensor and the quick capture interface configuration, and by appropriately setting their configuration registers.

3.1. Implementation of down-sampling

Down-sampling is achieved by setting the shape of the VSYNC and HREF signals to obtain a desired frame size, which will be QVGA (320×240) in our experiments. Moreover, it is necessary to select the zoom and scaling functionality and to set the horizontal and vertical scaling down coefficients by accessing the image sensor register set. By appropriate settings of the registers, in particular the Pixel Output Index (POIDX), the micro-controller in the OV9655 performs the scaling down by averaging the necessary pixels to get the desired frame size. The image sensor SCCB interface is directly handled by the API library available in the CITRIC camera SDK.

3.2. Implementation of cropping

The Digital Video Port interface of the image sensor is connected to the quick capture interface (Fig. 2). It outputs 10-bit data lines (D[9..0]), 3 synchronization signals (VSYNC, HREF, PCLK) and inputs the main clock MCLK. The acquisition of data from the sensor is initiated by transitions based on the state of the HREF and VSYNC signals, which are generated internally by the sensor. Once the quick capture interface is enabled, the capture sequence is activated by the assertion of the VSYNC signal, which indicates that a frame read-out is about to occur. The assertion of HREF causes the quick capture interface moving on to the active data capture state where a valid data line is captured under the control of the PCLK signal as depicted in Fig. 3. The image cropping is the selection of an area inside the whole image. This area is named "cropped window" and characterized by its position, width and height. The position

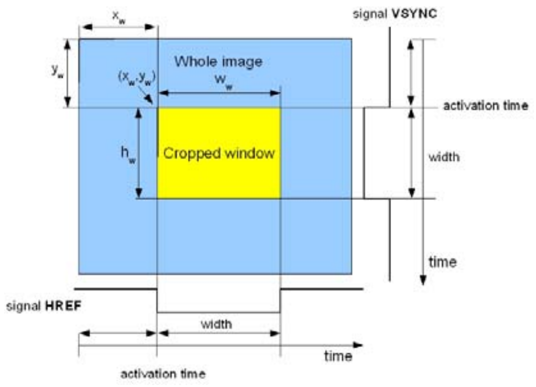

Figure 4. Implementing image cropping.

is the pixel coordinates of its upper left corner inside the whole image. The synchronization signal VSYNC indicates which sequence of lines has to be captured in a frame. Similarly, the signal HREF indicates which sequence of pixels has to be captured in each line. As seen in Fig. 4, the widths of the VSYNC pulse and HREF pulse correspond to the height and width of the cropped window, respectively. The time at which HREF and VSYNC are activated is related to the position of the upper left corner of the cropped window.

To perform cropping, it is necessary to change the sensor configuration so that it outputs pulses with modified position and width to select the cropped window. The configuration of the quick capture interface has to be changed accordingly to collect the new amount of data sent by the image sensor. The control of the quick capture interface and the DMA engine to perform cropping is more complex, since it requires a specific device driver. The abilities of some video capture devices to sample a subsection of an image and shrink it are exploited by Video4Linux API [16], but the newest kernel version [17], Linux-2.6.37, does not support the device driver for the CITRIC camera platform and for any ARM PXA270 based platforms equipped with the OV9655 image sensor. For instance, when we want to capture videos (successive frames), this sometimes produces multiple copies of the same frame. In our experiments, we were able to successfully capture static cropped frames to measure the savings in energy consumption when processing one frame, and project the increase in battery-life. Currently, a new kernel patch including an experimental device driver for the OV9655 sensor is under evaluation to capture successive cropped frames.

As will be detailed in Sections 5 and 6, hardware-level cropping provides significant savings in energy consumption and increase in battery lifetime. The reasons include the better use of the memory controller and the memory resources, a reduction in the amount of data that is moved from the image sensor to the main memory at each frame, and not occupying the main microprocessor with this task. One application to take advantage of cropping is the localized foreground object detection and tracking algorithm that is introduced in [12]. This application will be discussed in more detail in Section 4.

Another application wherein the cropping will be useful is surveillance scenarios with regions of interest (ROI). If a ROI is defined such that an alarm is triggered when an event is detected in this region, then we can make use of cropping to significantly increase energy efficiency.

4. Localized Detection and Tracking

Traditional tracking systems perform foreground object detection and tracking at each frame independently and in a sequential manner. This will henceforth be referred to as the *sequential method*. Casares et al. [12] introduced the *feedback method* that is a lightweight foreground object detection and tracking algorithm suitable for embedded platforms. In this method, feedback from the tracking stage is used to determine search regions, and perform detection and tracking in those regions instead of the whole frame. They showed that this provides significant savings in processing time, and thus increases idle state durations of cameras to increase the battery-life.

As described in Section 3.2, we present a method to crop the image at hardware-level so that energy efficiency is increased even further. The experimental results showing the decrease in energy consumption and the increase in battery-life are presented in Sections 5 and 6, respectively.

5. Processing time and Energy Savings

This section provides a quantitative comparison showing the advantages of performing hardware-level down-sampling and cropping at the micro-controller of the OV9655 sensor rather than processing whole frames and performing these tasks at software level on the main microprocessor of the camera board. In a CITRIC camera, the down-sampling of an image is performed through API software library. As mentioned in section 2.1, the OV9655 sensor has a IC board with a micro-controller. Delegating tasks such as down-sampling and cropping to this micro-controller avoids occupying the main ARM processor, and thus decreases its processing load, decreases processing time, and more importantly increases the battery-life of the embedded smart camera.

5.1. Grabbing a QVGA frame

Grabbing a frame in QVGA (320×240) resolution is the result of applying down-sampling to VGA images. As mentioned above, this operation was being done at software-level on the main ARM processor of the camera board. We have performed down-sampling at hardware-level at the micro-controller of the OV9655 sensor. Figures 5 (a) and (b) show QVGA images captured by the CITRIC camera using software and hardware down-sampling methods, respectively. At hardware-level, neighborhood averaging is used to down-sample. At software-level, instead of averaging, the API library routines drop repetitive information during the down-sampling. Thus, Fig. 5(a) is slightly sharper compared to Fig. 5(b).

Figure 5. QVGA images captured by (a) using the API software library down-sampling subroutines and (b) performing hardware-level down-sampling on the micro-controller of the OV9655.

Figure 6 shows the operating currents of the camera board while grabbing a QVGA frame using only the API software libraries and while performing the same task of down-sampling by hardware using the OV9655 and the quick capture interface. The grabbing takes 49.8 when using the API libraries, while it takes 30.78 ms when employing the hardware-level down-sampling at the micro-controller of the image sensor. This corresponds to 38.2% savings in grabbing time.

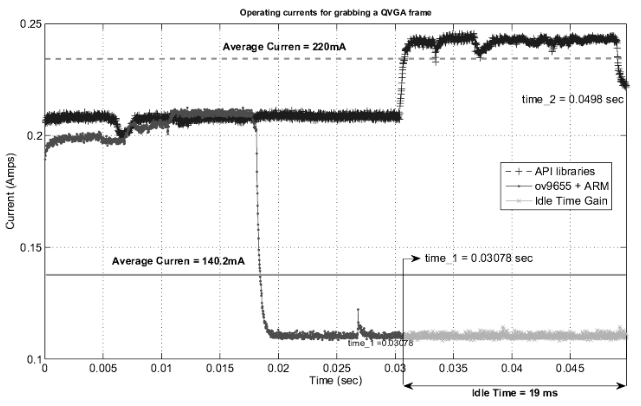

Figure 6. Operating currents of the camera board while grabbing a QVGA frame using the API sub-sampling subroutines and using the image micro-controller of the OV9655.

The dashed and solid lines in Fig. 6 show the average current levels when using software-level and hardware-level down-sampling, respectively. As can be seen, a 36.27% reduction in the average operating current is obtained when performing hardware-level down-sampling at the micro-controller of the OV9655 sensor. As shown in Table 1, this corresponds to 24.47% decrease in energy consumption. It should be noted that to compare the energy consumption of both scenarios, we sent the main ARM processor to IDLE state for $19ms$, so the time window is the same ($49.8ms$) for both cases (Fig. 6).

Table 1. Energy consumption when grabbing a QVGA frame using the API software libraries versus performing down-sampling at hardware-level.

Down-sampling method	Power (W)	Energy (mJ)
Software	1.1655	57.2
Hardware	0.7493	43.2
gain (%)	35.71%	24.47%

5.2. Foreground Detection Experiments

In this section, we will first compare the following: (i) obtaining QVGA images with software-level down-sampling and performing all processing (down-sampling and foreground object detection) on the main microprocessor of the camera board; (ii) performing down-sampling at hardware-level on the micro-controller of the OV9655 sensor, and performing foreground object detection at the main microprocessor. Figure 7 shows the operating current levels of the camera board when using these two approaches. As seen in this figure, collaborating with the image sensor, and hardware-level operations provide 43.7% savings in processing time as compared to the software-level down-sampling relying on the API libraries. In addition, it provides 23.94% savings in energy consumption.

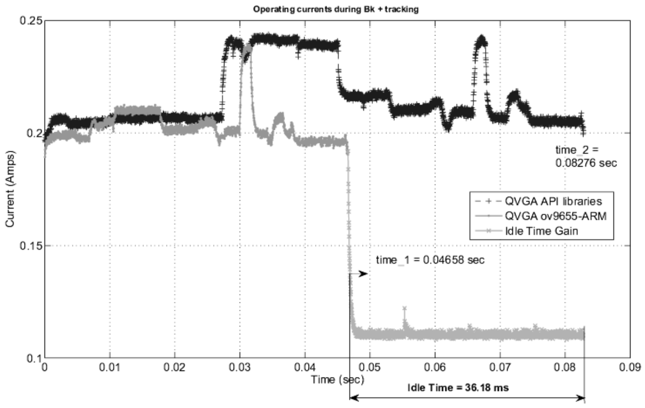

Figure 7. Operating currents when grabbing/buffering a frame and performing background segmentation using the API sub-sampling subroutines versus collaborating with the OmniVision OV9655.

Figures 8(b) and (c) show example foreground detection results when using the software-level and hardware-level down-sampling, respectively. As seen in Fig. 8(c), the output is slightly better when using the hardware-level down-sampling due to the slight blurring introduced by averaging neighboring pixels as discussed in section in section 5.1. This provides noise reduction, and thus better segmentation.

In this section, we also present savings in energy consumption when we perform hardware-level cropping and perform object detection only in the cropped region of a frame. As mentioned in Section 4, a lightweight object detection and tracking method is presented in [12], which employs feedback from the tracking stage to determine search

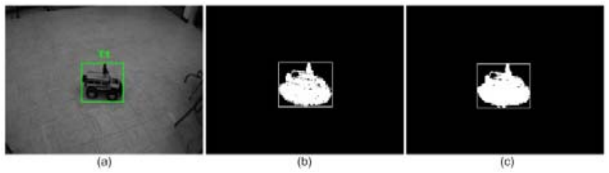

Figure 8. (a) Captured frame showing the bounding box, (b) background subtraction output on a frame grabbed by using the API software libraries to down-sample to QVGA resolution, (c) background subtraction output on a frame grabbed using hardware-level down-sampling.

regions, and perform detection and tracking in those regions only instead of the whole frame. This method provides significant savings in processing time, and thus allows us to increase idle state durations of cameras to increase the battery-life. Cropping the image at hardware-level, as described in Section 3.2, based on the search regions will increase energy efficiency even further. Figures 9(a) and (b) show example cropped images obtained by software and hardware-level cropping, respectively. Figure 10 shows the gains obtained by using the hardware capabilities of the OV9655 to crop search regions. It reduces the processing time by 55% and 62.8% compared to the software-based feedback method [12] and sequential method, respectively. In addition, the energy consumption is reduced by 44.92% and 54.136% compared to the software-based feedback method and sequential method, respectively. Table 2 summarizes the power and energy savings.

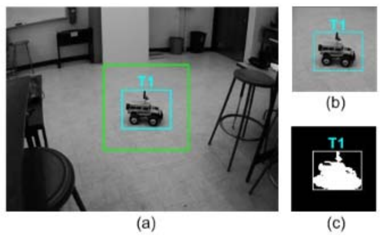

Figure 9. Area cropped (a) by software using the API libraries (b) by hardware using the micro-controller of the OV9655.

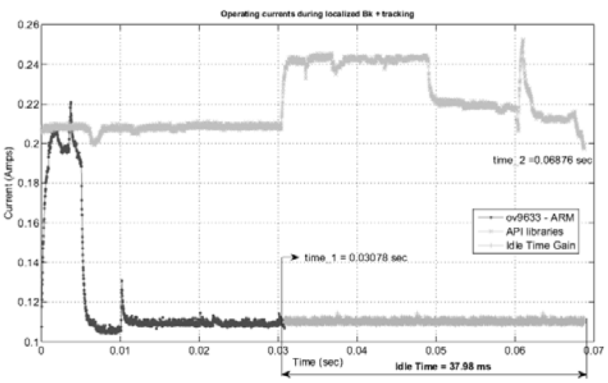

Figure 10. Operating currents when performing foreground object detection on cropped search regions obtained by software versus hardware level cropping.

Table 2. Energy consumption when grabbing and cropping a frame at software-versus hardware-level.

Method	Power (W)	Energy (mJ)	gain (%)
Sequential software-level	1.1143	92.23	–
Feedback software-level	1.1174	76.8	16.73%
Feedback hardware-level	0.6153	42.3	54.14%

When there are multiple objects in the scene, cropping can be performed in alternating frames.

6. Increase in Battery-life

We also projected the battery-life of the embedded smart camera for all three scenarios: (i) performing down-sampling at software-level, and performing the foreground object detection on the whole frame; (ii) performing down-sampling and cropping at software-level, and performing the foreground object detection on smaller search regions; (iii) performing down-sampling and cropping at hardware-level by exploiting the image sensor capabilities, and performing the foreground object detection on smaller search regions. We used a KIKUZO PLZ664WA (Electronic Load) to estimate the battery-life. It should be noted that the estimated lifetimes are based on the scenario in which there will always be an object to track in the scene, i.e. the scene will never be empty. Table 3 summarizes the battery lifetimes when using each of the three methodologies above. As can be seen, using the feedback method with hardware down-sampling and cropping will prolong the battery-life of the camera by 97% compared to the software-based feedback method presented in [12]. It will also provide a significant increase of 121.25% in the battery-life compared to the sequential method.

Table 3. Battery lifetime projection.

Method	Battery Lifetime (hours)	gain(%)
Sequential	7.48	-
Feedback Software-level	8.4	12.3%
Feedback by Hardware-level	16.55	121.25%

7. Conclusion

We have presented two methodologies to increase the energy-efficiency and the battery-life of an embedded smart camera by hardware-level operations when performing object detection. First, instead of performing down-sampling at software-level at the main microprocessor of the camera board, we perform this operation at hardware-level on the micro-controller of the OV9655 image sensor of a CITRIC camera. Second, we crop an image frame by using the HREF and VSYNC signals at the micro-controller of the OV9655, so that object detection can be performed only in the cropped search region. Hardware-level cropping can be combined with a feedback-based tracking method to increase energy efficiency even further.

Reduced amount of data that is moved from the image sensor to the main memory at each frame, better use of the memory resources and not occupying the main microprocessor with image down-sampling and cropping tasks, provide significant savings in energy consumption and battery-life. Experimental results show that, compared to software-level cropping, performing hardware-level cropping when tracking one object provides 97.2% increase in battery-life prolonging the life of the camera up to 16.55 hours. In addition, hardware-level down-sampling and cropping, and performing detection in cropped regions provide 54.14% decrease in energy consumption, and 121.25% increase in battery-life compared to performing software-level down-sampling and processing whole frame.

Acknowledgements

This work was supported partly by the National Science Foundation under grant CNS 0834753, and UNL Research Council Award. The authors would like to thank Fabio Parigi for his help during the battery-life analysis experiments.

References

[1] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach. Distributed embedded smart cameras for surveillance applications. *IEEE Computer*, 39: 68-75, 2006.

[2] P. Chen and et al., Citric: A low-bandwidth wireless camera network platform, *Proc. of the ACM/IEEE International Conference on Distributed Smart Cameras*, 2008.

[3] W.-C. Feng, W.-C. Feng, and M. L. Baillif. Panoptes: Scalable low-power video sensor networking technologies. *Proc. of the ACM Conference on Multimedia*, pp. 562-571, 2003.

[4] S. Fleck, F. Busch, P. Biber, and W. Strasser. 3d surveillancea distributed network of smart cameras for real-time tracking and its visualization in 3d. *Proc. of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, pp. 118, 2006.

[5] S. Hengstler, D. Prashanth, S. Fong, and H. Aghajan. Mesheye: A hybrid-resolution smart camera mote for applications in distributed intelligent surveillance. *Proc. of the International Symposium on Information Processing in Sensor Networks*, pp. 360-369, 2007.

[6] R. Kleihorst, A. Abbo, B. Schueler, and A. Danilin. Camera mote with a high-performance parallel processor for real-time frame-based video processing. *Proc. of the ACM/IEEE Int'l Conf. on Distributed Smart Cameras*, pp. 106-116, 2007.

[7] M. Quaritsch, M. Kreuzthaler, B. Rinner, H. Bischof, and B. Strobl. Autonomous multicamera tracking on embedded smart cameras. *EURASIP Journal on Embedded Systems*, 92827: 10, 2007.

[8] P. Kulkarni, D. Ganesan, P. Shenoy, The case for multi-tier camera sensor network. *Proc. of the ACM Workshop on Network and Operating System Support for Digital Audio and Video*, 2005.

[9] M. Rahimi and et al. Cyclops: In situ image sensing and interpretation in wireless sensor networks. *Proc. of the Int'l Conf. on Embedded Networked Sensor Systems*, pp. 192-204, 2005.

[10] B. Rinner, T. Winkler, W. Schriebl, M. Quaritsch, and W. Wolf. The evolution from single to pervasive smart cameras. *Proc. of the ACM/IEEE International Conf. on Distributed Smart Cameras*, 2008.

[11] A. Rowe, C. Rosenberg, and I. Nourbakhsh. A second generation low cost embedded color vision system. *Proc. of the IEEE Embedded Computer Vision Workshop in conjunction with the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 136, June 2005.

[12] M. Casares and S. Velipasalar, Resource-Efficient Salient Foreground Detection for Embedded Smart Cameras. *Proc. of the IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, 2010.

[13] Omnivision Technologies Inc. OV9655 Color CMOS SXGA (1.3MegaPixel) CAMERACHIP with OmniPixel Technology Datasheet, 2006.

[14] A. Kerhet, M. Magno, F. Leonardi, A. Boni, and L. Benini. A low-power wireless video sensor node for distributed object detection. *Journal of Real-Time Image Processing*, 2: 331–342, 2007.

[15] Intel PXA27x Processor Family Developers Manual. http://www.balloonboard.org/hardware/300/ds/PXA270-dev-manual.pdf

[16] Bill Dirks, Michael H. Schimek, Hans Verkuil and Martin Rubli. Video for Linux Two API Specification. Revision 0.24. http://v4l2spec.bytesex.org/spec/

[17] The Linux Kernel Archives. http://kernel.org