Energy-efficient Feedback Tracking on Embedded Smart Cameras by Hardware-level Optimization

Mauricio Casares, Senem Velipasalar
Department of Electrical Engineering
University of Nebraska-Lincoln
mauricio.casares@huskers.unl.edu
velipasa@engr.unl.edu

Paolo Santinelli, Rita Cucchiara
DII - University of Modena
and Reggio Emilia, Modena, Italy
paolo.santinelli@unimore.it
rita.cucchiara@unimore.it

Andrea Prati
DiSMI - University of Modena
and Reggio Emilia,
Reggio Emilia, Italy
andrea.prati@unimore.it

***Abstract*—Embedded systems have limited processing power, memory and energy. When camera sensors are added to an embedded system, the problem of limited resources becomes even more pronounced. In this paper, we introduce two methodologies to increase the energy-efficiency and battery-life of an embedded smart camera by hardware-level operations when performing object detection and tracking. The CITRIC platform is employed as our embedded smart camera. First, down-sampling is performed 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, instead of performing object detection and tracking on whole image, we first estimate the location of the target in the next frame, form a search region around it, then crop the next frame by using the HREF and VSYNC signals at the micro-controller of the image sensor, and perform detection and tracking only in the cropped search region. Thus, the amount of data that is moved from the image sensor to the main memory at each frame is optimized. Also, we can adaptively change the size of the cropped window during tracking depending on the object size. Reducing the amount of transferred data, better use of the memory resources, and delegating image down-sampling and cropping tasks to the micro-controller on the image sensor, result in significant decrease in energy consumption and increase in battery-life. Experimental results show that hardware-level down-sampling and cropping, and performing detection and tracking in cropped regions provide 41.24% decrease in energy consumption, and 107.2% increase in battery-life. Compared to performing software-level down-sampling and processing whole frames, proposed methodology provides an additional 8 hours of continuous processing on 4 AA batteries, increasing the lifetime of the camera to 15.5 hours.**

I. Introduction

The spread of embedded systems has increased enormously worldwide in recent years. Embedded systems equipped with smart camera sensors are employed in many different applications including environmental monitoring [1], stereo matching [2], as well as a plethora of applications related to video surveillance, such as foreground object detection [7], [5], [3], face detection [4], people detection [6] etc.

When a camera sensor is added to an embedded system, several problems arise due to the large amount of image data to be stored and processed. This impacts the overall efficiency, the memory requirements, the communication bandwidth and the energy consumption of the system. Most of the work related to camera-equipped embedded systems mainly focus on maximizing computational efficiency by optimizing the video processing algorithms. However, as the focus moves towards mobile applications [4], limited resources mandate consideration of energy consumption.

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. Battery-powered embedded smart cameras provide a lot of flexibility in terms of camera quantities and placement, however 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. Even though methods have been proposed for object detection and tracking with embedded systems, much less effort has been spent on developing applications that decrease the energy consumption of the embedded cameras. Reddy *et al.* [5] considered the problem of background bootstrapping with short sequences, taking into account the limited computational and memory resources in the targeted architecture. Tessens *et al.* [7] compared computational requirements and accuracy of two background subtraction methods that are based on single scan lines and rectangular regions of interest (ROI). Yet, the aforementioned works have not analyzed power consumption.

Fleck *et al.* [13] present a network of smart cameras for tracking people. They use 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.* [16] employ smart cameras with multiple DSP processors. Bramberger *et al.* [8] present a smart camera architecture with processing power of 9600 MIPS and 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 relatively high bandwidth for communication and powerful processing capabilities. Yet, they have high power consumption and are larger in size. Many embedded vision platforms have been developed more recently. The CMUcam2 [20] is a low-cost embedded camera

This work was supported partly by the National Science Foundation under grant CNS 0834753, and UNL Research Council Award.

978-1-4577-1707-9/11/$26.00 ©2011 IEEE

with 75MHz RISC processor and 384KB SRAM. Due to the limited memory and processing power, only low-level image processing can be performed. The camera mote introduced by Kleihorst *et al.* [15] has an 84MHz XETAL-II SIMD processor, and has a higher resolution. The Cyclops [18] and MeshEye [14] platforms have 7.3-MHz and 55-MHz processors, respectively. Thus, in both of these platforms the processing power is very limited. SensEye [17] is a multi-tier network of heterogeneous wireless nodes and cameras. Panoptes platform [12] hosts a 206-MHz processor, but has high energy consumption. Kerhet *et al.* [21] employ a hybrid architecture FPGA-MCU, where the processing load is distributed to increase the efficiency of the camera mote. Chen *et al.* [9] 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. Rinner *et al.* [19] present a comparison of various smart camera platforms.

Casares *et al.* [10] 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. Also, it was recently shown by Casares *et al.* [11] that performing hardware-level operations at the image sensor level significantly reduces the energy consumption of the embedded smart cameras. Yet, the presented results are based on experiments performed on a single frame with a fixed cropped window size. Continuous tracking of objects, with varying cropped window sizes, is not performed.

In this paper, we combine the feedback-based tracking method presented in [10] with hardware-level operations (specifically hardware-level down-sampling and cropping), and present a comprehensive tracking analysis. We perform hardware-level cropping with varying window sizes. We propose a solution that modifies the low-level kernel-driver structure of the camera, which resets the circular FIFO buffer of the camera. Our goal is to further increase the energy-efficiency and the battery-life by hardware-level operations when performing object detection and tracking. To achieve this two methods are presented: (i) rather than performing down-sampling and image cropping at the main microprocessor on the camera board, these operations are performed at the micro-controller of the OV9655; (ii) we first estimate the location of the target in the next frame, form a search region around it, and then crop the next frame by using the HREF and VSYNC signals at the micro-controller of the OV9655, and perform detection and tracking only in the cropped search region.

Performing these functions at hardware level provides multiple advantages including savings in processing time and a 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 as a result of the better use of the memory controller and the memory resources, and by delegating image down-sampling and cropping tasks to the micro-controller on the image sensor.

We have compared the energy consumption of the following when detecting and tracking an object: (i) performing down-sampling by software using the CITRIC API libraries, and processing whole frames; (ii) implementing down-sampling and cropping by software using the CITRIC API libraries, and performing detection and tracking in cropped regions; (iii) implementing down-sampling and cropping by hardware using the micro-controller of the image sensor, and performing detection and tracking only in cropped regions. The experimental results are presented, which show the savings in processing time and energy consumption, and the gain in the battery-life of the CITRIC camera when using the feedback from the tracking stage, and performing down-sampling and cropping operations at hardware-level.

II. The Embedded Smart Camera Platform

The wireless embedded smart camera platform employed in our experiments is a CITRIC mote [9] shown in Fig. 1. It consists of a camera board and a wireless mote. 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 microprocessor is connected to a 64 MB of SDRAM and 16MB of NOR FLASH. The image sensor is an OmniVision OV9655. An embedded Linux system runs on the camera board. Attached to the camera board is a TelosB mote with a maximum data rate of 250 Kbps.

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

A. Frame Capture Operation

The PXA270 microprocessor incorporates peripheral functions to handle the image sensor. These are the Intel Quick Capture Interface (QCI), the direct memory access (DMA) controller and the I2C interface. The QCI provides a connection between the processor and the image sensor. It is able to acquire data and control signals, and performs the appropriate data formatting prior to routing the data to memory using DMA. The I2C interface is used to access the configuration registers set of the image sensor. The QCI interface requires a parallel data-bus interface, two synchronization signals HREF and VSYNC for frame timing and a pixel clock for basic timing. HREF is the "line valid" signal, and VSYNC is the "frame valid" signal. The timing signals VSYNC and HREF, provided by the sensor, activate and reset the quick capture

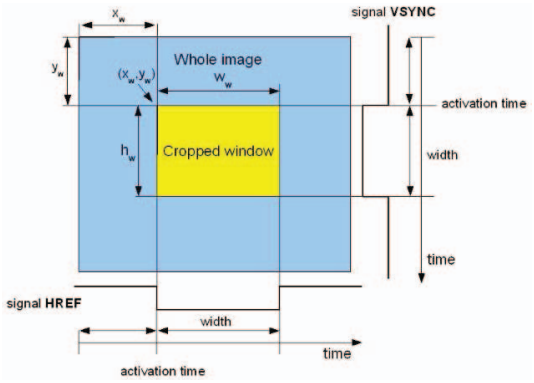

Fig. 2. Implementing image cropping.

interface that can be configured to provide an interrupt at the end of each line and each frame as described in [11].

III. Image Scaling and Cropping

The main goal of our work 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 search region obtained from the tracking stage. 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 transferred data. 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. Down-sampling, scaling and cropping operations are accomplished by changing the hardware registers of the OV9655. 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 as explained in the OV9655 operation manual, and described in [11].

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 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 shown in Fig. 2.

To perform down-sampling and grab a frame in QVGA resolution, the VSYNC and HREF are set so that the whole information acquired by the sensor is used. 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.

As will be detailed in Sections V and VI, hardware-level cropping provides significant savings in energy consumption and increase in battery lifetime. One application to take advantage of hardware-level cropping is the localized foreground object detection and tracking algorithm introduced in [10]. This application will be discussed in more detail in Section IV.

The original kernel version running on the CITRIC camera was an optimized and patched kernel imported from the original Linux kernel 2.6.9. The image sensor of the CITRIC camera is handled by a device driver that is obtained by customizing the "Video For Linux One" driver for the OV9650 image sensor and ARM processor so that the driver can work for the newest OV9655 image sensor. As described in [11], the image sensor is equipped with two different interfaces shown in Fig. 3. The Serial Camera Control Bus (SCCB) interface is used to program the sensor behavior. The Digital Video Port interface provides a connection between the sensor and the quick capture interface to acquire data and control signals, and performs the appropriate data formatting prior to routing the data to memory.

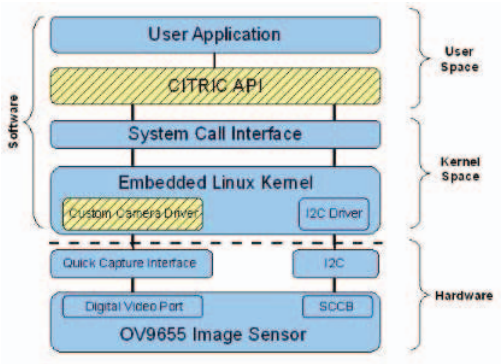

Fig. 3. Software architecture handling the CITRIC camera board.

The software to perform image cropping and image down-sampling consists of several functions. These functions are used in "live" or "run-time" mode and some of them are employed to dynamically change the position and the size of the cropped window inside the whole image. The functions for the reconfiguration of the quick capture interface and the control of the DMA engine are based on the Video 4 Linux Standard IOCTL and allow us to collect the right amount of data sent by the image sensor. Additionally, some of these functions are used to clean the frame circular buffer of the device driver. The tracking algorithm employs these functions to achieve time synchronization capabilities which allow us to perform tracking in "run-time". The function for the reconfiguration of the image sensor register set works in user space and it has been added to the API library available in the CITRIC camera SDK. The other functions work in kernel space and have been implemented as new API IOCTL provided by the Linux device driver for the image sensor as shown in Fig. 4.

IV. 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*. We have presented the *feedback method* [10], which is a lightweight foreground object detection and tracking algorithm suitable for embedded platforms. In this

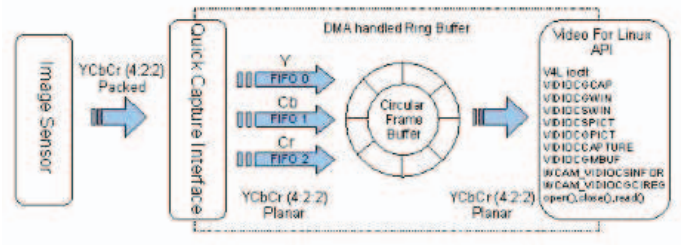

Fig. 4. Camera Driver Internal architecture.

method, feedback from the tracking stage is used to determine search regions for the following frame, and perform detection and tracking only in those regions instead of the whole frame. We have shown that this provides significant savings in processing time, and thus increases idle state durations of cameras to increase the battery-life.

We have also presented [11] an analysis of the energy consumption when implementing hardware-level down-sampling and cropping, and performing detection only in the cropped region. Yet, the results presented in [11] are based on experiments performed on a single frame with a fixed cropped window size. Continuous tracking of objects, with varying cropped window sizes, is not performed.

In this paper, the feedback method [10] is used to determine a search region in the following frame. Then, we crop the next image at hardware-level as described in Section III. After cropping, the detection and tracking are performed on the search areas as seen in Fig. 5. The experimental results showing the decrease in energy consumption and the increase in battery-life are presented in Sections V and VI, respectively.

To actually implement the tracking system, the original CITRIC-kernel-2.6.9 has been updated to version 2.6.23, and the Linux device driver for the image sensor has been modified. The kernel of the CITRIC camera was not capable of dynamically changing the size of the cropped regions from frame to frame. Thus, to overcome this issue, we have customized the existing device driver of the OV9655 contained in the CITRIC-kernel-2.6.23 so that we can use it to dynamically crop regions in run-time. The operating system architecture of the CITRIC camera is presented in Fig. 3. The striped yellow boxes are the modules that have been modified to dynamically change the size of cropped window for tracking purposes.

V. Savings in Energy Consumption

In this section, we provide a quantitative comparison showing the advantages of performing hardware-level down-sampling and cropping at the micro-controller of the OV9655 sensor for tracking purposes rather than processing whole frames and performing these tasks at software level on the main micro-processor of the camera board.

We will present savings in energy consumption when we perform hardware-level down-sampling and cropping, and use the feedback method for object detection and tracking. As stated in [10], the feedback method provides significant savings in processing time, and thus allows us to increase idle state durations of cameras to increase the battery-life. As described in Section IV, in the feedback method, information from the tracking stage is used to determine search regions in the next frame so that detection and tracking can be performed only in these regions instead of the whole frame. Figure 5 shows a sequence of frames in which a remote-controlled car is tracked. Figure 5(a) shows a QVGA frame grabbed during the tracking of the remote-controlled car. Whole frames are grabbed until the displacement of the target is computed from two consecutive frames. Then, the location of the target is estimated at the following frame. A search region of size $2w \times 2h$ is formed around this location, where w and h are the width and height of the bounding box in the current frame, respectively. The details can be found in [10]. Then, the following frame is cropped to the search region at hardware level, and the detection and tracking are performed only in the cropped region as depicted in Fig. 5(b). To show the movement of the car, and the changing cropped window, a small red circular reference point is highlighted on the cropped frame sequence.

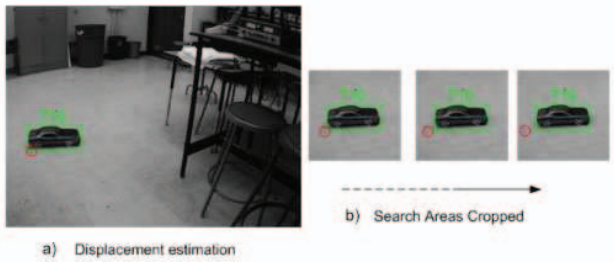

Fig. 5. (a) Last QVGA frame captured while computing pixel displacement of the tracked object; (b) Search regions cropped at hardware level.

Before presenting the energy consumption analysis during feedback-based tracking combined with hardware-level cropping, we will first compare the following two scenarios on a single QVGA size frame to separately show the contribution of hardware-level down-sampling to the savings: (i) obtaining QVGA images with software-level down-sampling, and performing all processing (down-sampling, foreground object detection and tracking) 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 and tracking at the main microprocessor. The operating currents of the camera board while using these approaches are presented in Fig. 6. Even though collaborating with the image sensor and hardware-level operations slightly prolong the processing time per frame by 22 ms, they considerably decrease the energy consumption of the camera by 27.38% as presented in Table I.

We now present savings in energy consumption when we use the feedback method for object detection and tracking, and perform hardware-level cropping. The aforementioned scenarios analyzed for QVGA resolution, are now performed in a reduced search region cropped by software or hardware-level operations. The software-based feedback method [10] grabs a frame in VGA resolution, down-samples it and crops the search region all by software. On the other hand, the hardware-level method uses the capabilities of the OV9655

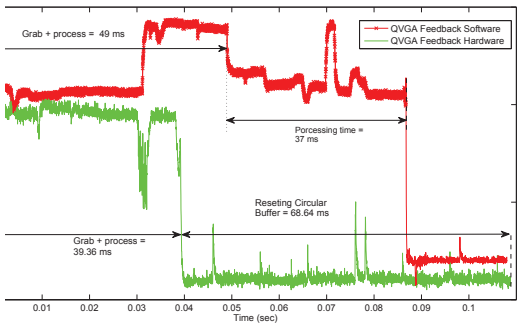

Fig. 6. Operating currents when (i) obtaining QVGA images with software-level down-sampling, and performing all processing on the main microprocessor ; (ii) performing down-sampling at hardware-level on the micro-controller of the OV9655 sensor, and performing foreground object detection and tracking at the main microprocessor.

TABLE I
ENERGY CONSUMPTION WHEN GRABBING A QVGA FRAME AT SOFTWARE- VERSUS HARDWARE-LEVEL, AND PERFORMING DETECTION AT THE MAIN MICROPROCESSOR.

	QVGA		
Method	Power (W)	Energy (mJ)	gain (%)
Software-level down-sampling	1.0415	112.5	–
Hardware-level down-sampling	0.751	81.7	27.38%

to down-sample, and then crop the search region. Having the search regions, foreground detection and tracking tasks are performed only in those regions at the main micro-processor of the CITRIC camera. Figure 7 shows the operating current of the camera board when (i) using the feedback method implemented entirely at software level; and (ii) applying hardware-level operations for cropping and down-sampling.

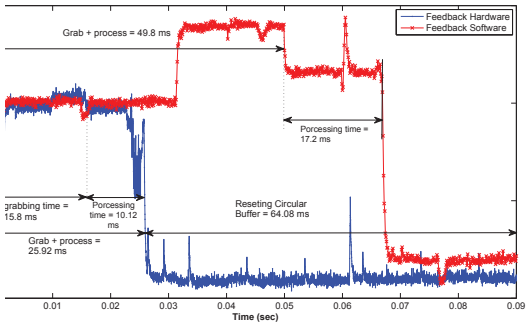

Fig. 7. Operating currents when performing foreground object detection an tracking on cropped search regions obtained by software- versus hardware-level cropping.

Even though the processing time increases by 23 ms when cropping a search region of 100×100 pixels at hardware-level and processing it, using the hardware capabilities of the OV9655 provides 28.3% decrease in energy consumption of compared to software-level cropping and processing. Different scenarios are summarized in Table II presenting the energy consumption and savings when processing a single frame.

TABLE II
ENERGY CONSUMPTION WHEN GRABBING AND CROPPING A SEARCH REGION (100X100) AT SOFTWARE- VERSUS HARDWARE-LEVEL AND PERFORMING DETECTION AT THE MAIN MICROPROCESSOR.

	100x100 Search Area		
Method	Power (W)	Energy (mJ)	gain (%)
Sequential software-level	1.0415	112.5	–
Feedback software-level	1	92.23	18.02%
Feedback hardware-level	0.719	66.1	41.24%

We have also performed an experiment, wherein we tracked a remote-controlled car continuously for 3 seconds, and changed the size of the cropped window once every second. We measured the energy consumption during this period. Figure 8 shows the operating current of the camera board for different scenarios during 1-second portion of this 3-second experiment. As explained in Section III, the circular buffer is reset when performing hardware-level cropping, which slightly increases the processing time of a frame. However, feedback method combined with hardware-level cropping provides 29.4% and 37% decrease in energy consumption compared to the software-based feedback method and sequential method, respectively. Table III summarizes the power and energy consumptions, and savings.

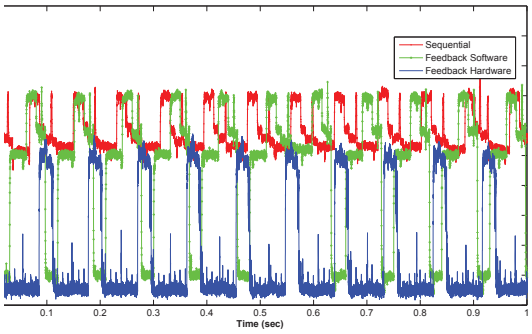

Fig. 8. Operating currents when performing foreground object detection and tracking during 1-second time interval.

TABLE III
ENERGY CONSUMPTION WHEN PERFORMING DETECTION AND TRACKING DURING A 3-SECOND TIME INTERVAL AT SOFTWARE-VERSUS HARDWARE-LEVEL.

Method	Power (W)	Energy (J)	gain (%)
Sequential software-level	1.1422	3.4273	–
Feedback software-level	1.0203	3.0608	10.7%
Feedback hardware-level	0.7203	2.1609	37%

When there are multiple objects in the scene, we need to form multiple search regions, and crop multiple windows. In this case, hardware-level cropping can still be performed for one window per frame, and different windows for different objects can be cropped at alternating frames.

VI. Increase in Battery-life

We also projected the battery-life of the cameras for the following scenarios: (i) Sequential method: performing down-sampling at software-level, and detecting and tracking objects in the whole frame; (ii) Software-level feedback method: performing down-sampling and cropping at software-level, and detecting and tracking objects in smaller search regions; (iii) Hardware-level feedback method: performing down-sampling and cropping at hardware-level by exploiting the image sensor capabilities, and detecting and tracking objects in smaller search regions. We used the characteristics curves provided by the manufacturer of the batteries. When there is one car in the scene, the average currents drawn are 0.2162 A, 0.1926 A and 0.1345 A for scenarios (i), (ii) and (iii), respectively. The projected battery lifetimes and energy savings are summarized in Table IV. As can be seen, when feedback method is combined with hardware-level operations (scenario (iii)), the battery life increases to 15.5 hours, and it provides 84.52% and 107.2% increase in battery-life compared to scenarios (i) and (ii), respectively. It should be noted that the projected battery lifetimes are based on the scenario, where there will always be an object to track in the scene, i.e. the scene will never be empty.

TABLE IV
Battery lifetime projection.

Method	Battery Lifetime (hours)	gain(%)
Sequential method	7.48	-
Software-level Feedback Method	8.4	12.3%
Hardware-level Feedback Method	15.5	107.2%

VII. 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 and tracking. 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. Moreover, rather than performing object detection and tracking on the whole frame, we estimate the location of the target in the next frame, form a search region around it, crop the next frame by using the HREF and VSYNC signals at the micro-controller of the OV9655, and perform detection and tracking only in the cropped search region.

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 84.52% increase in battery-life prolonging the life of the camera up to 15.5 hours. In addition, hardware-level down-sampling and cropping, and performing detection and tracking in cropped regions provide 41.24% decrease in energy consumption, and 107.2% increase in battery-life compared to performing software-level down-sampling and processing whole frame.

References

[1] R. Pon, M.A. Batalin, J. Gordon, A. Kansal, D. Liu, M. Rahimi, L. Shirachi, Y. Yu, M. Hansen, W.J. Kaiser, M. Srivastava, G. Sukhatme, and D. Estrin. Networked infomechanical systems: a mobile embedded networked sensor platform. *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks*, pp. 376–381, 2005.

[2] S. Gehrig, F. Eberli, and T. Meyer. A real-time low-power stereo vision engine using semiglobal matching, Computer Vision Systems. *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, v. 5815, pp. 134-143, 2009,

[3] M. Casares, S. Velipasalar and A. Pinto. Light-weight Salient Foreground Detection for Embedded Smart Cameras. *Computer Vision and Image Understanding*, vol. 114, issue 11, pp. 1223–1237, 2010.

[4] V. Kianzad, S. Saha, J. Schlessman, G. Aggarwal, S. S. Bhattacharyya, W. Wolf, and R. Chellappa. An architectural level design methodology for embedded face detection. *Proc. of the 3rd IEEE/ACM/IFIP International Conference on Hardware/software Codesign and System Synthesis*, pp. 136-141, 2005.

[5] V. Reddy, C. Sanderson, B.C. Lovell, and A. Bigdeli. An efficient background estimation algorithm for embedded smart cameras. *Third ACM/IEEE International Conference on Distributed Smart Cameras*, pp. 1-7, 2009.

[6] Zaihong Shuai, Songhwai Oh, and Ming-Hsuan Yang. Traffic modeling and prediction using camera sensor networks. *Proc. of the Fourth ACM/IEEE International Conf. on Distributed Smart Cameras*, pp. 49-56, 2010.

[7] L. Tessens, M. Morbee, W. Philips, R. Kleihorst, and H. Aghajan. Efficient approximate foreground detection for low-resource devices. *Proc. of the Third ACM/IEEE International Conf. on Distributed Smart Cameras*, pp. 1-8, 2009.

[8] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach. Distributed embedded smart cameras for surveillance applications. *IEEE Computer*, 39:6875, 2006.

[9] 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.

[10] M. Casares and S. Velipasalar. Resource-Efficient Salient Foreground Detection for Embedded Smart Cameras br Tracking Feedback. *Proc. of the 2010 IEEE International Conf. on Advanced Video and Signal Based Surveillance*, pp. 369–375, 2010.

[11] M. Casares, P. Santinelli, S. Velipasalar, A. Prati and R.Cucchiara. Energy-efficient Object Detection and Tracking on Embedded Smart Cameras by Hardware-level Operations. *Proc. of the Seventh IEEE Workshop on Embedded Computer Vision*, 2011.

[12] 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.

[13] 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 Conf. on Computer Vision and Pattern Recognition Workshop*, pp.118, 2006.

[14] 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.

[15] 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. 106116, 2007.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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 IEEE Conf. on Computer Vision and Pattern Recognition*, page 136, June 2005.

[21] 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*, vol. 2, pp. 331–342, 2007.