Large-scale Intrusion Detection with Low-cost Multi-camera wireless image sensors

Congduc Pham
University of Pau, LIUPPA Laboratory
Email: congduc.pham@univ-pau.fr

***Abstract*—We present a low-cost multi-camera wireless image sensor platform based on commercial off-the-shelf components implementing an intrusion detection mechanism for large-scale surveillance applications. We first show the performance of the image sensor then present how a criticality-based scheduling approach can be adapted and integrated into the image sensor platform taking into account the real hardware constraints and capabilities. We then demonstrate by simulating large-scale configurations that although hardware constraints limit the maximum intrusion detection rate per node, the usage of a low-cost multi-camera system provides much higher detection quality than denser single-camera system working at a higher image capture rate.**

***Index Terms*—Low-cost image sensors, wireless image sensor networks, multi-camera, intrusion detection**

I. Introduction

A Wireless Image Sensor Networks (WISN) consists of a set of sensor nodes equipped with miniaturized image cameras. In this paper, we are more particularly interested in WISN for surveillance applications that have very specific needs due to their inherently critical nature associated to security. Usually, high coverage and energy preservation are orthogonal properties and obviously a tradeoff needs to be found. In randomly deployed sensor networks, image sensor node's fields of view (FoV) can be redundant leading to overlaps among the monitored areas. In this case, a common tradeoff is to define a subset of the deployed nodes to be active while the other nodes can sleep. The result is a schedule of the activity of image sensor nodes in such a way that maximizes the deployment area coverage as well as the network lifetime.

For intrusion detection applications, we proposed in [1], [2] that nodes with a high redundancy level can increase their image capture rate because if they run out of energy they can be replaced by other nodes that can view the same area. For a given image node i, a set of nodes that can cover a significant part of node i's FoV is call a cover-set. Therefore we proposed to link in a criticality-based model the image capture rate to both the application's criticality level and the node's number of cover-sets. A low criticality level indicates that the application does not require a high image frame capture rate while a high criticality level does. Previous works on intrusion detection/mission-critical surveillance applications such as [3], [4], [5], [6], [7], [8] mostly focused on coverage and energy optimizations without taking explicitly into account the application's criticality. In this paper, we show how this criticality-based scheduling mechanism can be adapted and integrated into the real image sensor platform, taking into account limited capture rate allowed by the hardware and the image processing tasks.

There are a number of image sensor boards available or proposed by the very active research community on image and visual sensors: Cyclops [9], MeshEyes [10], Citric [11], WiCa [12], SeedEyes [13], Eye-RIS [14], Panoptes [15], CMUcam3&FireFly [16], [17], CMUcam4 and CMUcam5/PIXY [17], iMote2/IMB400 [18] and other specific solutions for ultra low-power platforms such as [19], [20]. Though highly interesting, all these platforms and/or products mostly rely on ad-hoc development of the visual part (i.e. development of a camera board with dedicated micro-controller to perform a number of processing tasks) or use very powerful micro-controller/Linux-based platforms. In addition, they usually lack an efficient image encoding and compression scheme adapted to very low-bandwidth radio. Our motivations in building our own image sensor platform for research on image sensor surveillance applications are:

1) to have an off-the-shelf solution so that anybody can reproduce the hardware and software components: we use an Arduino-based solution for maximum flexibility in programming and design; we use a simple, affordable external camera to get raw image data. We can therefore easily extend to a multi-camera system to enhance the detection quality as shown in the paper.
2) to integrate an efficient image compression scheme running on host micro-controller (no additional nor dedicated micro-controller) which addresses the problem of resource limitations of sensor nodes, as memory size, processor speed, battery capacity and low-bandwidth radio, and which produces a packet stream tolerant to packet losses.
3) to be flexible enough to allow the development and the integration of additional control mechanisms such as image change detection (for intrusion detection) and advanced node activity scheduling. For instance, we will describe how we integrated our criticality-based image sensor scheduling method [1], [2] to enable large-scale intrusion detection applications.

The image sensor we built works with raw 128x128 8-bbp gray scale image which can be compressed with various quality factors for reducing the bandwidth usage and end-to-end delay of image communication over the multi-hop network path. An intrusion detection mechanism based on simple-differencing technique shows very good results while adding no cost in the image processing. In addition, one original feature with respect to previous works on image sensors is the generalization from a single-camera system to a multiple-camera system. With a 3-camera system and wide-angle lenses omnidirectional sensing can almost be achieved.

Obviously, the maximum image capture rate allowed by a sensor platform is an important parameter for mission-critical intrusion detection. From the real image sensor platform we can measure the maximum image capture and intrusion detection rate as well as the transmission costs. Then, by using simulation of the adapted criticality-based scheduling on large-scale configurations, this article shows that a low-cost multi-camera system can successfully detect intrusions and provides a high level of responsiveness.

The rest of the article is organized as follows. Section II describes the generic image sensor components (hardware components and image processing) and the main performance measures. Then Section III presents the integration of a criticality-based image node scheduling mechanism to enable large-scale deployment and Section IV will show by simulations that the proposed multi-camera system can provide efficient intrusion detection. We conclude in Section V.

II. Multi-camera image sensor

A. Hardware components

We use Arduino boards with the CMOS uCamII camera from 4D systems. The uCamII is shipped with a 56^o angle of view lens but we can also use 76^o and 116^o lenses. The uCam is connected to the Arduino board through an UART interface at 115200 bauds. The uCamII is capable of providing both raw and JPEG bit streams but we are not using this last feature as it is impossible from the delivered JPEG bit stream to build a packet stream tolerant to packet losses. As a result, we retrieve raw 128x128 8-bpp grey scale images from the uCamII then we operate image compression on the Arduino board. We actually have two versions of our image sensor: one is based on the Arduino Due board and the other on the Arduino MEGA2560. The Arduino Due is a micro-controller board based on the Atmel SAM3X8E ARM Cortex-M3 running at 84MHz with 96KB of SRAM memory. The MEGA2560 features an ATmega2560 at 16Mhz and has only 8KB of SRAM memory. Although the very limited amount of memory of the MEGA was a challenge for implementing image processing tasks, we found that the MEGA board consumes much more energy than the Due for all operations, in addition to higher processing time. Therefore in this paper, we will only consider the Due platform. The radio module can either be an IEEE 802.15.4 radio provided by a Digi XBee S1 module for short-range communications or a Libelium SX1272 LoRa$^{\text{TM}}$ module for long-range communication. However, the particular case of LoRa$^{\text{TM}}$transmission will not be treated in this paper. The XBee is connected through a serial port at 125000 bauds as communication between the XBee and Arduino is not reliable at 115200 bauds given the board clock frequency of 84MHz (or 16MHz for the MEGA).

Then from the 1-camera system it is not difficult to have a multiple cameras system as the Arduino Due has 4 UART ports. One port is used for connection to the XBee module if short-range communication is used, so the 3 others are available for 3 uCamII cameras. Figure 1(left) shows our Arduino Due connected to 3 uCamII cameras. 76^o and 116^o lenses can be mounted on the uCamII, in addition to the 56^o lens shipped with the uCamII. Figure 1(right) compares the FoV of the 3 lenses.

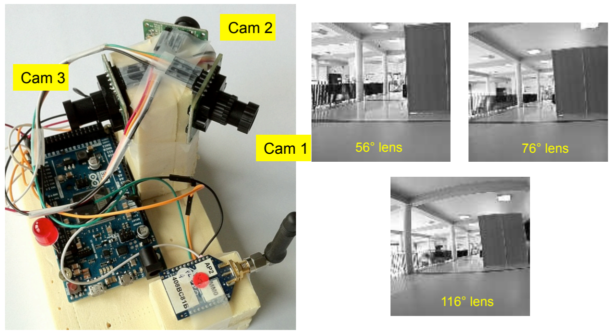

Fig. 1. A 3-camera system on the Arduino Due

The cameras are set at 120^o from each other to provide near disk coverage property with 116^o lenses. Only one camera can be active at a given time and the image change detection is performed on each camera with the corresponding reference image.

B. Image processing tasks

We describe below the two image processing tasks implemented on the image sensor.

1) Image change detection: We implemented an image change detection mechanism based on ”simple-differencing” of pixel: each pixel of the image from the uCam is compared to the corresponding pixel of a reference image, taken previously at startup of the image sensor. Little modifications in luminosity due to the camera imperfect white balance is taken into account by computing the mean luminosity difference between the captured image and the reference image. Additionally, if no image change occurs during 5 minutes, the sensor takes a new reference image to take into account light condition changes. Many tests have been performed to validate the intrusion detection mechanism and we were able to systematically detect a single person intrusion at 25m without any false alert as shown in Fig. 2.

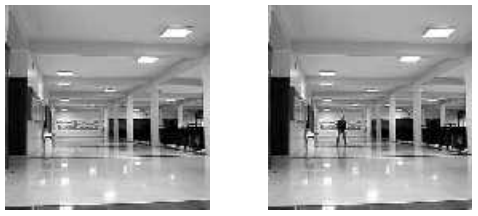
Fig. 2. Left: reference image; Right: intruder detected

2) Image compression method: We use an optimized encoding scheme proposed in [21] to obtain a packet-tolerant encoded bit stream. It features the 2 following key points:

1) Image compression must be carried out by independent block coding in order to ensure that data packets correctly received at the sink are always decodable.
2) De-correlation of neighboring blocks must be performed prior to packet transmission by appropriate interleaving methods in order to ensure that error concealment algorithms can be efficiently processed on the received data.

The compression scheme is a JPEG-like coder and operates on 8x8 pixel blocks with advanced optimizations on data computation to keep the computational overhead low. The combination of the fast JPEG-like proposed encoder with an optimized block interleaving method [22] allows for an efficient tuning, the so-called Quality Factor (Q), of the compression ratio/energy consumption trade-off while maintaining an acceptable visual quality in case of packet loss. The code has been ported to Arduino with little modifications. Fig. 3 shows the original raw 128x128 image taken with the image sensor and encoded with various quality factors: Q=90 (high quality), Q=50 (medium quality) and Q=10 (low quality). The total size of the compressed image, the number of generated packets and the PSNR compared to the original image are shown. For multi-camera system, note that when images need to be transmitted (upon intrusion), each camera can be configured with a different image quality factor if necessary. Fig. 3 also shows the impact of packet losses (20% and 40%).

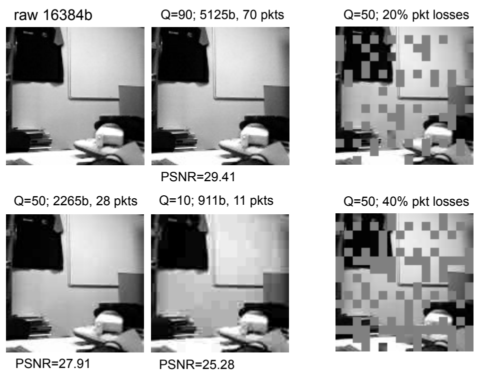

Fig. 3. 128x128 image taken by the image sensor, various quality factor.

We set the maximum image payload per packet to 90 bytes (in practice, the produced packet size will vary according to the packetization process) because 9 bytes need to be reserved in the 802.15.4 payload for header information.

C. Performance measures

1) Image processing: Fig. 4 shows the encoded image size with the compression ratio and the number of produced packets for various quality factors. Column A shows the image encode time which is quite constant. Column B shows the "encode+pkt time" which is the overhead of the image encoding process including the encoding itself and the packetization stage, but without transmission. The time to read the raw image data from the uCam is also shown in column R (1512ms) and it actually does not depend much on the uCam-Arduino connection baud rate (here 115200 bauds) because the limitation is mainly due to memory read operations from the Arduino UART ring buffer. R+B represents the latency between the snapshot taken by the camera and the time all the packets of the encoded image are produced (once again without transmission).

If we take into account the transmission overhead shown in column C, column D shows the "encode+pkt+transmission time". The packetization and the transmission tasks are performed in a row for each packet. Values in column B and column D have been globally measured and can be used to get column C which is the time taken globally for transmitting the produced packets: more packets means higher transmission time. If we use a quality factor of 50, the total time between the snapshot taken by the uCam and the end of the transmission of the image is 1512+879=2391ms. Column E shows the image sensor cycle time with transmission of image packets.

		N	R	A	B	C = D - B	D	E=R+D	F
Quality Factor Q	size in bytes (compression ratio)	number of packets (with MSS=90)	reading time from ucam	encode time	encode + pkt time	transmission time (deduced)	encode + pkt + transmission time	cycle time, with transmission	rcv time at the sink
90	5125 (3.2)	70	1512	512	782	539	1321	2833	799
80	3729 (4.4)	48	1512	511	704	384	1088	2600	599
70	2957 (5.5)	37	1512	519	686	304	990	2502	447
60	2552 (6.4)	32	1512	509	662	263	925	2437	390
50	2265 (7.2)	28	1512	500	646	233	879	2391	349
40	2024 (8.1)	25	1512	516	657	207	864	2376	317
30	1735 (9.5)	21	1512	516	649	177	826	2338	278
20	1366 (12)	17	1512	518	638	140	778	2290	231
10	911 (18)	11	1512	516	628	93	721	2233	177

Fig. 4. Cycle time measured on the Due-based platform as a function of the image compression ratio. All times are in ms.

To quantify the cost of the intrusion detection mechanism, we measured the time to get data from the uCam when the "simple-differencing" method is included and when it is not. We did not observe any difference: the time to read data from the uCam and perform the pixel comparison is still 1512ms.

2) Transmission: We measured the time between 2 image packets sent by the image sensor (when sending image packets as fast as possible) with a promiscuous IEEE 802.15.4 sniffer connected to the wireshark packet analysis tool. We found that the mean time between 2 image packets is between 11ms and 12ms. These measures are very consistent with previous measures shown in Fig. 4. For instance, if we take column C divided by the number of packets, C/N, we find about 8ms. If we take $(B - A)/N$ for quality factor above 50 we find 3ms to 4ms. Therefore the inter-arrival time at packet sniffer can be decomposed as about 8ms for the transmission time and 3ms to 4ms for the packetization time.

In Fig. 4, column F showed the receive time measured at a sink which will decode and display the image. The receive time represents the elapsed time between the first packet received and the last packet received which already takes into account the transmission time at the source. Fig. 5 then shows the 1-hop image display latency which is the elapsed time between the snapshot at the source image sensor and the display of the image at the sink 1-hop away (column R+B+F). The smaller the latency, the more responsive is the system. Since the time to transmit a packet is not very large, we can actually see that having Q up to 70 is not very penalizing.

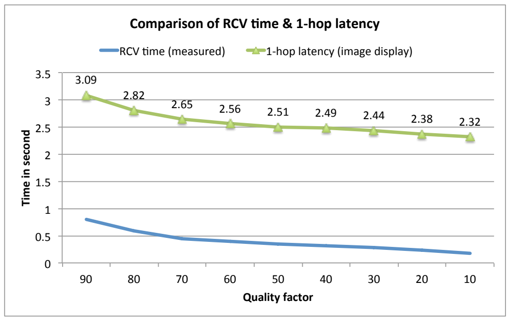

Fig. 5. Image received time and 1-hop image display latency

For multi-hop transmission using relay nodes, our previous work on performance measures of various relay platforms can be read in [23], [24]. We found for instance that the MicaZ platform provides the smallest relaying time (15ms for a 100-byte packet) among various commonly found platforms: TelosB, iMote2 and Arduino MEGA/Due. In multi-hop transmission, the 1-hop image display latency shown in Fig. 5 will then be increased by the relay time at each node multiplied by the number of intermediate relay nodes. As the relay time is quite small compared to the other image sensor's time overheads, the cost of multi-hop relaying in not high.

3) Energy consumption: To measure the energy consumption we inserted additional power consumption by toggling a led to better identify the various phases of the image sensor operations. For all the energy tests, the transmitted image was encoded using a quality factor of 50 which generates between 25 and 29 packets. Fig. 6 shows an entire cycle of camera sync, camera config, data read, data encode and packetization with transmission on the Due.

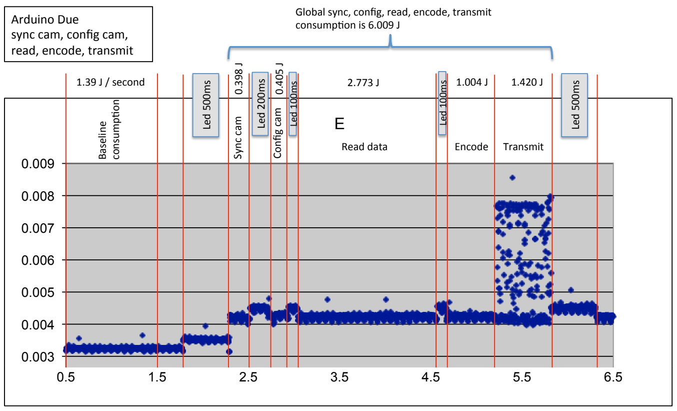

Fig. 6. Energy consumption for the Arduino Due

The current baseline energy consumption of the Due has been measured at 1.39J/s. This value is however not relevant as we did not implement any advanced power saving mechanisms such as putting the micro-controller in deep sleep mode or lower frequency, or performing ADC reduction, nor powering off the radio module. It is expected that the baseline consumption can be much further decreased with more advanced power management policy. Most interestingly, after removing the energy consumed by the led, we found that an entire cycle for image acquisition, encoding and transmission consumes about 6J. The largest consumed energy part on the Due comes from polling the serial line to get the image data from the uCam (through the system serial buffer). The encoding process actually consumes less than half that amount of energy. The cost of periodic image change detection, without encoding and transmission is similar to the "Read data" cost. Therefore, we found that the "Global sync, config, read&compare" consumption is 3.571J. If we use a 1200mAh 9V battery, i.e. 38880J, the image sensor can perform 10887 cycles of intrusion detection when assuming very low energy consumption in idle mode.

III. Large scale intrusion detection system

The criticality-based scheduling described in [1], [2] links the image capture rate of a node to the application's criticality level and the node's number of cover sets. We show in this section how the model's parameters can be adapted to the real image platform constraints.

A. Review of criticality-based node scheduling

The criticality-based model uses concave and convex curves as illustrated in Fig. 7. These type of curves have the following interesting properties for mission-critical applications:

- a concave curve has most projections of x values on the y-axis close to 0 (Fig. 7 box A). Such curve could represent "low criticality" applications that do not need high image capture rate;
- a convex curve where most projections of x values on the y-axis are close to the maximum frame capture rate (Fig. 7 box B). Such curve could represent "high criticality" applications that need high image capture rate.

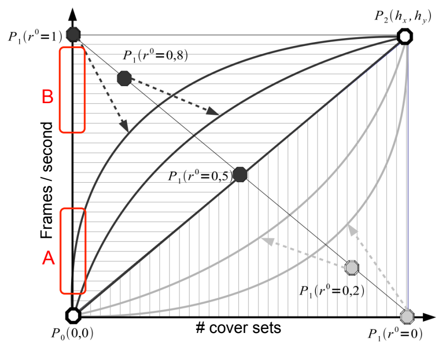

Fig. 7. The Behavior curve functions

We use Bezier curves to model the 2 application classes. 3 points can define a convex (high criticality) or concave (low criticality) curve: $P_0(0,0)$ is the origin point, $P_1(b_x, b_y)$ is the behavior point and $P_2(h_x, h_y)$ is the threshold point where h_x is the highest number of cover-sets and h_y is the maximum frame capture rate determined by the sensor node hardware capabilities. As illustrated in Fig. 7, by moving the behavior point P_1 inside the rectangle defined by P_0 and P_2, we are able to adjust the criticality level using either a convex or a concave form. P_1 therefore defines a criticality level r^0 which is between 0 and 1, 1 being the highest criticality level which requires fast image capture rate. Fig. 8(left) shows the image capture rate curve for a criticality level of 0.8, a maximum image capture rate set to 3fps and a maximum number of cover-sets of 12 (nodes with higher number of cover sets will only consider 12 cover sets), i.e. $P_2(h_x, h_y) = (12, 3)$.

With this criticality-based scheduling approach image nodes with high number of cover-sets will have a higher frame capture rate, therefore increasing their probability to detect intruders. These nodes can therefore act as sentry nodes even though there is no explicit election procedure. Fig. 8(right) shows some nodes with their number of cover-sets, cs, and their associated image capture rate, fps. The black node in the center has the highest capture rate in its neighborhood and will therefore have higher probability of detecting intruders.

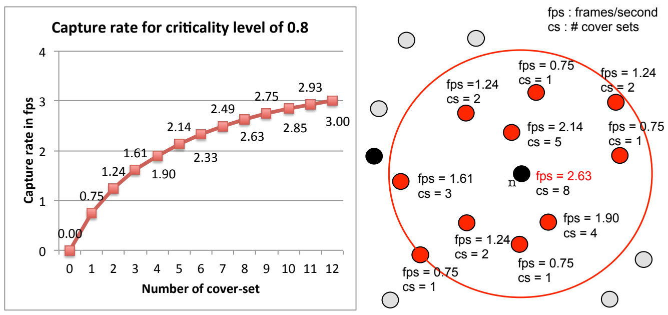

Fig. 8. Left: Capture rate according to criticality level and number of cover-sets. Right: Image node capture rate under criticality-based scheduling.

B. *Adapting the model to real image sensor capabilities*

As described above, for a given criticality level, the maximum image capture rate, the maximum number of cover-sets and the number of cover-sets give the current image capture rate for a node. For a single-camera node the FoV can be represented by a triangle and more complex computations are required to find overlaps and build the node's cover-sets [25]. With our multi-camera system, by using 3 cameras with 116^o lenses, the coverage is almost omnidirectional therefore disk coverage can be assumed which reduces the cost of sensor deployment and greatly simplify the coverage computation to determine the redundancy level of each sensor. For the maximum image capture rate, the developed image sensor needs a time between each snapshot of about 0.220s (cam sync time) + 0.220 (cam config time) + 1.512s (time to read image data and perform the intrusion detection)=1.952s giving a maximum image capture rate of about 0.52fps. Note that as only one camera can be active at a given time, the multiple cameras are activated sequentially (e.g. round robin manner for instance). Fig. 9 shows the adapted criticality model using $P_2 = (8, 0.52)$. The top figure plots for a criticality level is 0.8 (high criticality level), while the bottom figure shows the 0.2 case (low criticality level). In both graphs, the right-most axis shows the time between 2 snapshots in seconds.

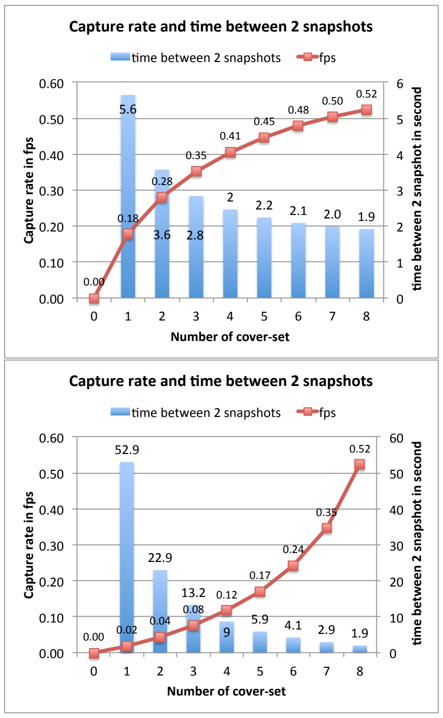

Fig. 9. Criticality model adapted to the image sensor hardware

Our image sensor platform can be configured dynamically at runtime by setting (i) its number of cover-sets, (ii) the maximum number of cover-sets and (iii) the criticality level. The maximum number of cover-sets taken for the criticality curve can be configured between 6 and 12. We set the number of maximum cover-sets to 8. A higher value (such as 12 in Fig. 8) provides much larger inter-snapshot time when the number of cover-sets is small. Using a smaller value (such as 8 or 6) has the advantage to give more significant difference in inter-snapshot time when the number of cover-sets is varied. According to the surveillance application profile, the maximum number of cover-sets can be defined prior to deployment, or can even be set dynamically during the image sensor operation.

IV. SIMULATING LARGE-SCALE SCENARIOS

The simulation model we built for our previous contributions on criticality-based scheduling is developed under the OMNET++/Castalia framework. The model integrates the image encoding scheme and allows for "real" image packet transmissions under the communication stack and physical radio models (IEEE 802.15.4 in non-beacon CSMA mode). The simulation configuration file indicates the image file that will be transmitted upon intrusion detection. An implementation of a geographical routing protocol enables multi-hop transmission of image packets from an image sensor source to a predefined sink node which will then decode and display the received image. We use a dynamic criticality approach where the entire network starts with a low criticality level, e.g. 0.2, and upon intrusions, alert messages will set the criticality level at a higher value, e.g. 0.8, for a given period of time before going back to normal operation. We randomly deployed 80 sensor nodes in a 400mx400m area and reproduced our image sensor features and constraints:

- camera angle of view of 76^o & 116^o,
- depth of view of 35m,
- 1 & 3 cameras/sensor (disk coverage is then assumed)
- maximum frame capture rate of 0.52fps,
- maximum number of cover-set is 8,
- 128x128 image (the one of Fig. 3)
- Quality Factor Q is set to 50, 2265B, 28 packets
- time before image data can be processed is 1.512s,
- encoding time is 500ms (Due's performance),
- inter-packet generation time is 11ms.

Random intrusions are sequentially introduced in the simulation model and move at a 5m/s velocity. Nodes can detect an intrusion if the intruder is covered by their FoV at the time of the image capture. Upon intrusion detection by a node, the stealth time of the intrusion is recorded and the node will broadcast an alert message to its 1-hop neighbors so that they can increase their criticality level themselves. Then the node sends the image packets to the sink. Fig. 10 shows the screenshot of the simulation with one image sent by node 46 to node 3 (sink). Node 72 serves as relay node under the GPSR routing protocol.

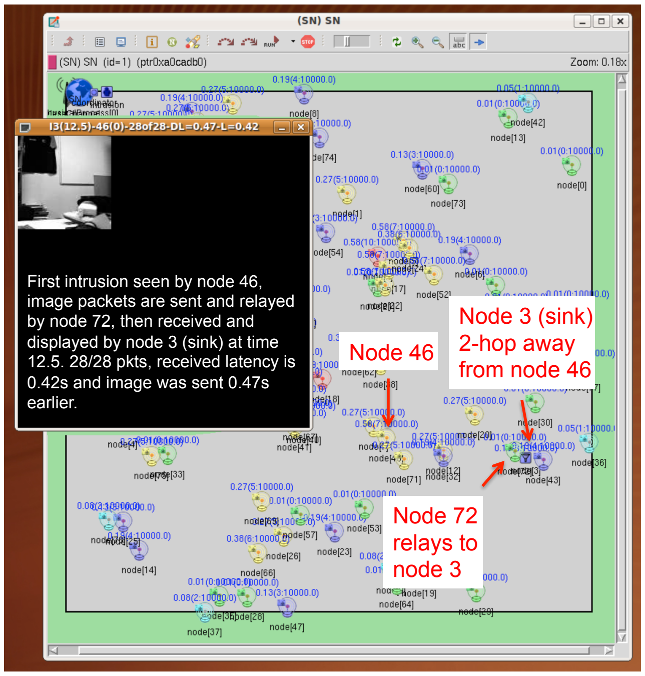
First intrusion seen by node 46, image packets are sent and relayed by node 72, then received and displayed by node 3 (sink) at time 12.5. 28/28 pkts, received latency is 0.42s and image was sent 0.47s earlier.
(SN) SN
(SN) SN (id=1) (ptr0xa0cadb0)
Zoom: 0.18x
I3(12.5)-46(0)-28of28-DL=0.47-L=0.42
Node 46
Node 3 (sink) 2-hop away from node 46
Node 72 relays to node 3

Fig. 10. Screenshot of the simulation environment with image transmission

In Fig. 11 we compare the coverage of a 80 x 1-uCamII system (top-left) to a 80 x 3-uCamII system (top-right) and to a 240 x 1-uCamII system (bottom-left) with 76^o lenses and when the image sensors are randomly deployed in an 400mx400m area. The depth of view of the cameras has been set to 35m. The FoV in red is the one of camera 1, for both 1-camera and 3-camera systems. The blue is for camera 2 and the green for camera 3, in the 3-camera system. We can see that the coverage is greatly improved, at a much lower cost than having 3 times more full sensor boards (bottom-left). With 116^o lenses, using 3 cameras can almost provide omnidirectional vision. Fig. 11(bottom-right) shows the coverage using such lenses.

The cameras are activated in a round-robin manner, according to the criticality-based scheduling mechanism. We have not studied whether the round-robin activation is adapted or not as the intrusion appear randomly. It is possible to change the camera activation policy but this is out of the scope of the current paper.

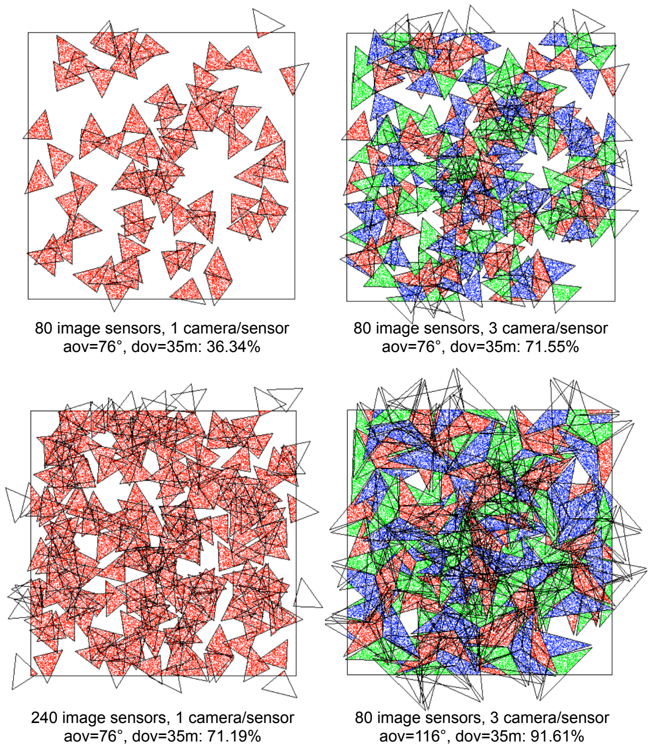

Fig. 11. Comparison of coverage by various image sensor systems

We evaluate the performance of the intrusion detection system with the stealth time which is the time during which an intruder can travel in the field without being seen. We showed in [26] that the criticality-based node scheduling approach can increase both the surveillance quality and the network lifetime compared to a static frame capture rate approach. In this paper, we will validate (i) that the adapted image capture rate model corresponding to the real developed hardware constraints can still offer high intrusion detection quality and, (ii) the efficiency of the multi-camera system even with simple round-robin camera activation. In Fig. 12 we plot the mean stealth time for the 80-node 1-camera 76^o configuration but using the configuration of [26] where the maximum image capture rate was arbitrarily set to 3fps and the maximum number of cover-sets was 12 (see Fig. 8).

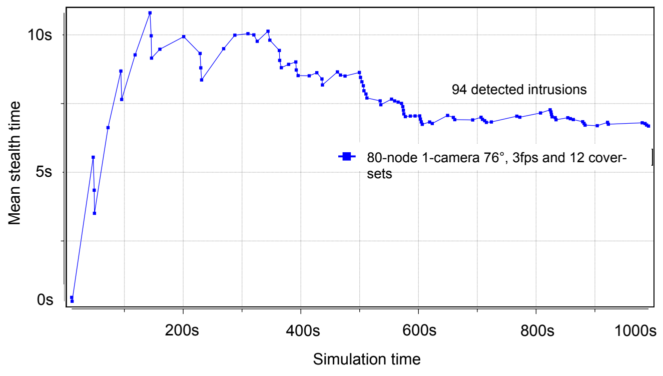

Fig. 12. Mean stealth time in the 80-node using $P_2 = (12, 3.0)$

Now, we use the adapted frame capture rate model corresponding to the real hardware capabilities (see Fig. 9). In Fig. 13 we plot the mean stealth time for the 80-node configuration (1-camera 76^o, 3-camera 76^o & 3-camera 116^o) and the 240-node configuration (with 1-camera 76^o system).

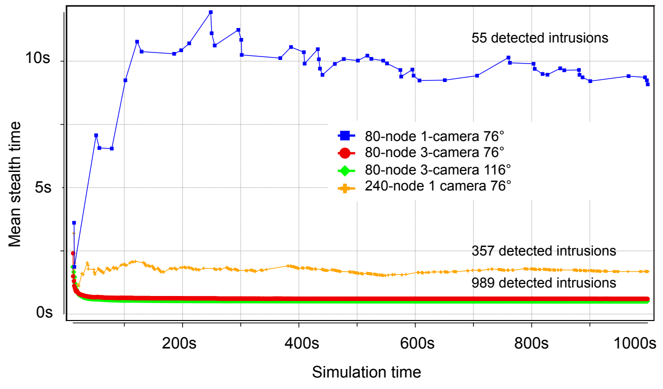

Fig. 13. Mean stealth time in the 80-node/240-node configurations

The first observation is that a lower image capture rate does increase the mean stealth time in the 80-node 1-camera 76^o configuration when compared to Fig. 12. Now, if we compare the 1-camera system to the 3-camera system, while the 80-node 1-camera system needs about 10s to detect the intruder as it moves across the area, we can see that the 3-camera system can greatly reduce the mean stealth time of intruders: both the 76^o and the 116^o setup can detect most intrusions in less than 0.5 second when it appears in the area although the maximum image capture rate is only 0.52fps. The 116^o setup shows a slightly smaller stealth time than the 76^o setup but we can see that having 3 cameras greatly increases the responsiveness of the system (in addition to the number of detected intrusions), even with 76^o cameras. We also show the performance of the 240-node 1-camera system that has the same number of cameras than the 80-node 3-camera system. We can see that although the mean stealth time can decrease to about 1.75s to detect the intrusion, this setup can not achieve the same detection latency than a 3-camera system with 3 times smaller number of nodes.

In Fig. 14 we show a 40-node configuration (1-camera 76^o, 3-camera 76^o & 3-camera 116^o) and a 120-node configuration with 1-camera system. The 40-node 1-camera 76^o needs about 20s while the 40-node 3-camera can detect most intrusions in less than 1s.

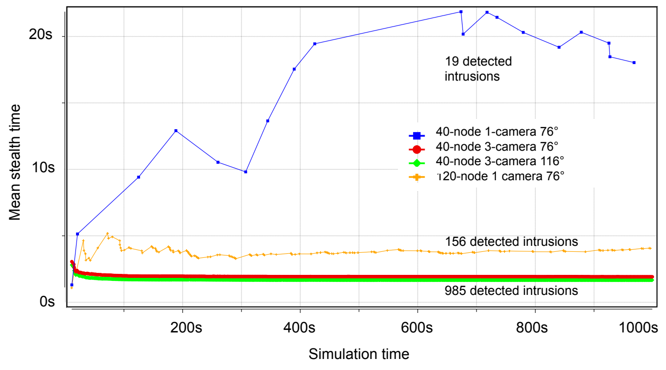

Fig. 14. Mean stealth time in the 40-node/120-node configurations

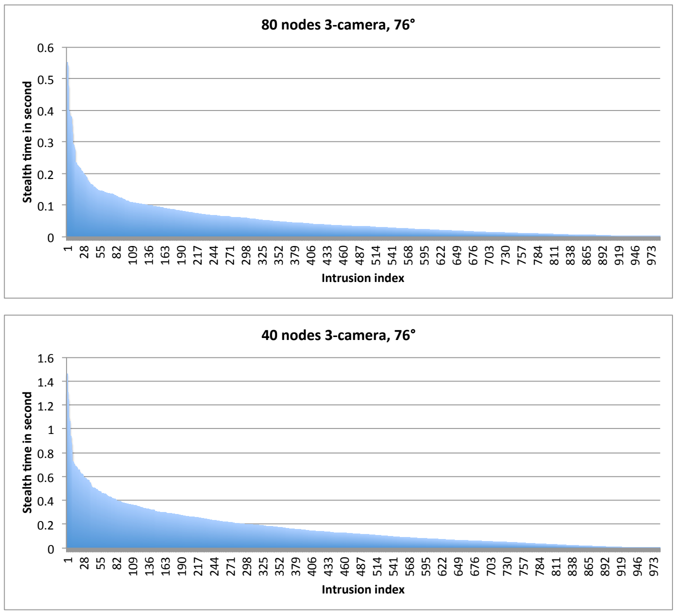

Fig. 15. Stealth time of intrusions in 40-node and 80-node configurations

In Fig. 15(top) we show the stealth time of all intrusions of the 80-node 3-camera system 76^o and compare it to a 40-node 3-camera 76^o system in Fig. 15(bottom). While the mean stealth time in both case is very small (see Fig. 13 and Fig. 14), we can better differenciate them in terms of stealth time distribution. For instance, the 40-node case has more than 30% of stealth time greater than 0.2s and more than 8% greater than 0.4s. The 80-node case has less than 3% of stealth time greater than 0.2s and about 0.3% greater than 0.4s. However, the detection quality is still very promising with the 40-node 3-camera system when compared to 1-camera configurations with 40, 80, 120 and 240 nodes. The results clearly show that using multiple cameras can provide fast and efficient intrusion detection with a small number of image nodes.

V. Conclusions

We presented a generic low-cost image sensor built from commercial off-the-shelf electronic components for maximum flexibility and availability to the research community. The performance measures of the image sensor to operate the image encoding and transmission process show that the 1-hop image display latency can be less than 2.5s with a reasonable image quality. We generalized the single-camera system to a multi-camera system for better coverage at a low cost. With such multi-camera system which can provide almost omnidirectional sensing, we integrated and adapted to real hardware constraints a criticality-based image node scheduling mechanism to enable large-scale deployment of image sensors. Simulations of large-scale scenarios showed that fast and efficient intrusion detection can be realized with a small number of multi-camera image nodes.

Acknowledgment

We would like to thank V. Lecuire from CRAN laboratory for the image encoding code that has been included in both the simulation model and the image sensor.

References

[1] A. Makhoul, R. Saadi, and C. Pham, "Risk management in intrusion detection applications with wireless video sensor networks," in *IEEE WCNC*, 2010.

[2] C. Pham, A. Makhoul, and R. Saadi, "Risk-based adaptive scheduling in randomly deployed video sensor networks for critical surveillance applications," *Journal of Network and Computer Applications*, vol. 34, pp. 783–795, 2011.

[3] J. Kostrzewa, W. H. Meyer, W. A. Terre, S. Laband, and G. W. Newsome, "Use of a miniature infrared cots sensor in a several military applications," in *Proc. SPIE*, vol. 4743, 2002, pp. 141–149.

[4] O. Dousse, C. Tavoularis, and P. Thiran, "Delay of intrusion detection in wireless sensor networks," in *ACM MobiHoc*, 2006.

[5] Y. Zhu and L. M. Ni, "Probabilistic approach to provisioning guaranteed qos for distributed event detection," in *IEEE INFOCOM*, 2008.

[6] E. Freitas, T. Heimfarth, C. Pereira, A. Ferreira, F. Wagner, and T. Larsson, "Evaluation of coordination strategies for heterogeneous sensor networks aiming at surveillance applications," in *IEEE Sensors*, 2009.

[7] M. Keally, G. Zhou, and G. Xing, "Watchdog: Confident event detection in heterogeneous sensor networks," *Real-Time and Embedded Technology and Applications Symposium, IEEE*, 2010.

[8] R. Kozma, L. Wang, K. Iftekharuddin, E. McCracken, M. Khan, K. Islam, and R. Demirer, "Multi-modal sensor system integrating cots technology for surveillance and tracking," in *IEEE Radar Conference*, May 2010, pp. 1030–1035.

[9] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava, "Cyclops: In situ image sensing and interpretation in wireless sensor networks," in *ACM SenSys*, 2005.

[10] S. Hengstler, D. Prashanth, S. Fong, and H. Aghajan, "Mesheye: A hybrid-resolution smart camera mote for applications in distributed intelligent surveillance," in *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, April 2007, pp. 360–369.

[11] P. Chen, P. Ahammad, C. Boyer, S.-I. Huang, L. Lin, E. Lobaton, M. Meingast, S. Oh, S. Wang, P. Yan, A. Yang, C. Yeo, L.-C. Chang, J. Tygar, and S. Sastry, "Citric: A low-bandwidth wireless camera network platform," in *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, Sept 2008, pp. 1–10.

[12] R. Kleihorst, A. Abbo, B. Schueler, and A. Danilin, "Camera mote with a high-performance parallel processor for real-time frame-based video processing," in *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*, Sept 2007, pp. 69–74.

[13] Evidence Embedding Technology, "Seed-eye board, a multimedia wsn device. http://rtn.sssup.it/index.php/hardware/seed-eye," accessed 20/12/2013.

[14] . Rodrguez-Vzquez, R. Domnguez-Castro, F. Jimnez-Garrido, S. Morillas, J. Listn, L. Alba, C. Utrera, S. Espejo, and R. Romay, "The eye-ris cmos vision system," in *Analog Circuit Design*, H. Casier, M. Steyaert, and A. Van Roermund, Eds. Springer Netherlands, 2008, pp. 15–32.

[15] W.-C. Feng, E. Kaiser, W. C. Feng, and M. L. Baillif, "Panoptes: Scalable low-power video sensor networking technologies," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 1, no. 2, pp. 151–167, May 2005.

[16] A. Rowe, D. Goel, and R. Rajkumar, "Firefly mosaic: A vision-enabled wireless sensor networking system," in *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*, Dec 2007, pp. 459–468.

[17] Evidence Embedding Technology, "Cmucam: open source programmable embedded color vision sensors. http://www.cmucam.org/," accessed 19/12/2014.

[18] S. Paniga, L. Borsani, A. Redondi, M. Tagliasacchi, and M. Cesana, "Experimental evaluation of a video streaming system for wireless multimedia sensor networks," in *Proceedings of the 10th IEEE/IFIP Med-Hoc-Net*, 2011.

[19] L. Gasparini, R. Manduchi, M. Gottardi, and D. Petri, "An ultralow-power wireless camera node: Development and performance analysis," *Instrumentation and Measurement, IEEE Transactions on*, vol. 60, no. 12, pp. 3824–3832, 2011.

[20] D. M. Pham and S. M. Aziz, "Object extraction scheme and protocol for energy efficient image communication over wireless sensor networks," *Computer Networks*, vol. 57, no. 15, pp. 2949 – 2960, 2013.

[21] V. Lecuire, L. Makkaoui, and J.-M. Moureaux, "Fast zonal dct for energy conservation in wireless image sensor networks," *Electronics Letters*, vol. 48, no. 2, 2012.

[22] C. Duran-Faundez and V. Lecuire, "Error resilient image communication with chaotic pixel interleaving for wireless camera sensors," in *Proceedings of ACM Workshop on Real-World Wireless Sensor Networks*, 2008.

[23] C. Pham, V. Lecuire, and J.-M. Moureaux, "Performances of multi-hops image transmissions on ieee 802.15.4 wireless sensor networks for surveillance applications," in *IEEE WiMob*, 2013.

[24] C. Pham, "Communication performances of ieee 802.15.4 wireless sensor motes for data-intensive applications: A comparison of waspmote, arduino mega, telosb, micaz and imote2 for image surveillance," *Journal of Network and Computer Applications*, vol. 46, no. 0, pp. 48 – 59, 2014.

[25] C. Pham and A. Makhoul, "Performance study of multiple cover-set strategies for mission-critical video surveillance with wireless video sensors," in *IEEE WiMOB*, 2010.

[26] C. Pham, "Network lifetime and stealth time of wireless video sensor intrusion detection systems under risk-based scheduling," in *IEEE ISWPC*, 2011.