Scheduling randomly-deployed heterogeneous video sensor nodes for reduced intrusion detection time

Congduc Pham

University of Pau, LIUPPA Laboratory,
Avenue de l'Université - BP 1155
64013 PAU CEDEX, FRANCE
Congduc.Pham@univ-pau.fr
http://web.univ-pau.fr/~cpham

Abstract. This paper proposes to use video sensor nodes to provide an efficient intrusion detection system. We use a scheduling mechanism that takes into account the criticality of the surveillance application and present a performance study of various cover set construction strategies that take into account cameras with heterogeneous angle of view and those with very small angle of view. We show by simulation how a dynamic criticality management scheme can provide fast event detection for mission-critical surveillance applications by increasing the network lifetime and providing low stealth time of intrusions.

Keywords: Sensor networks, video surveillance, coverage, mission-critical applications

1 Introduction

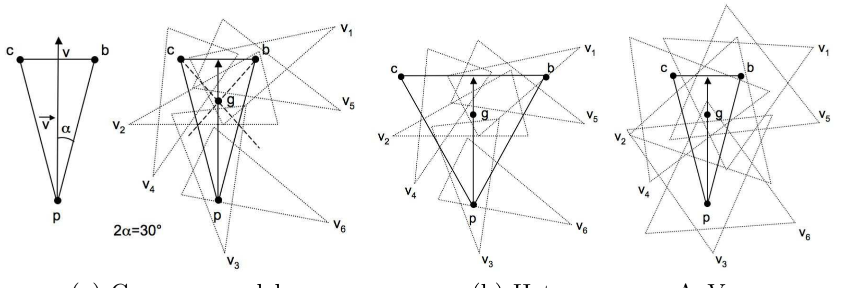
The monitoring capability of Wireless Sensor Networks (WSN) make them very suitable for large scale surveillance systems. Most of these applications have a high level of criticality and can not be deployed with the current state of technology. This article focuses on Wireless Video Sensor Networks (WVSN) where sensor nodes are equipped with miniaturized video cameras. We consider WVSN for mission-critical surveillance applications where sensors can be thrown in mass when needed for intrusion detection or disaster relief applications. This article also focuses on taking into account cameras with heterogeneous angle of view and those with very small angle of view.

Surveillance applications [1–5] have very specific needs due to their inherently critical nature associated to security . Early surveillance applications involving WSN have been applied to critical infrastructures such as production systems or oil/water pipeline systems [6, 7]. There have also been some propositions for intrusion detection applications [8–11] but most of these studies focused on coverage and energy optimizations without explicitly having the application's criticality in the control loop which is the main concern in our work. For instance, with video sensors, the higher the capture rate is, the better relevant events could be detected and identified. However, even in the case of very mission-critical applications, it is not realistic to consider that video nodes should always capture at their maximum rate when in active mode. The notion of cover set has been introduced to define the redundancy level of a sensor nodes that monitor the same region. In [12] we developed the idea that when a node has several covers, it can increase its frame capture rate because if it runs out of energy it can be replaced by one of its cover sets. Then, depending on the application's criticality, the frame capture rate of those nodes with large number of cover sets can vary: a low criticality level indicates that the application does not require a high video frame capture rate while a high criticality level does. According to the application's requirements, an R^0 value that indicate the criticality level could be initialized accordingly into all sensors nodes prior to deployment.

Based on the criticality model we developed previously in [12], this article has 2 contributions. The first contribution is an enhanced model for determining sensor's cover sets that takes into account cameras with heterogeneous angle of view and those with very small angle of view. The performance of this approach is evaluated through simulation. The second contribution is to show the performance of the multiple cover sets criticality-based scheduling method proposed in [12] for fast event detection in mission-critical applications. The paper is then organized as follows: Section 2 present the coverage model and our approach for quickly building multiple cover sets per sensor. In section 3 we quickly present the dynamic criticality management model and then present the main contribution of this paper that focuses on fast event detection in section 4. We conclude in section 5.

2 Video sensor model

A video sensor node v is represented by the FoV of its camera. In our approach, we consider a commonly used 2-D model of a video sensor node where the FoV is defined as a triangle (pbc) denoted by a 4-tuple $v(P, d, \overrightarrow{V}, \alpha)$. Here P is



(a) Coverage model

(b) Heterogeneous AoV

Fig. 1. Coverage model.

the position of v, d is the distance pv (depth of view, DoV), $\overrightarrow{V}$ is the vector representing the line of sight of the camera's FoV which determines the sensing direction, and α is the angle of the FoV on both sides of $\overrightarrow{V}$ (2α can be denoted as the angle of view, AoV). The left side of figure 1(a) illustrates the FoV of a video sensor node in our model. The AoV (2α) is 30^o and distance bc is the linear FoV which is usually expressed in ft/1000yd or millimeters/meter. By using simple trigonometry relations we can link bc to pv with the following relation $bc = \frac{2\sin\alpha}{\cos\alpha}.pv$. We define a cover set $Co_i(v)$ of a video node v as a subset of video nodes such that: $\bigcup_{v' \in Co_i(v)}(v'$'s FoV area) covers v's FoV area. $Co(v)$ is defined as the set of all the cover sets $Co_i(v)$ of node v.

One of the first embedded camera on a wireless sensor hardware is the Cyclops board designed for the CrossBow Mica2 sensor [13] which is advertized to have an AoV of 52^o. Recently, the IMB400 multimedia board has been designed for the Intel Mote2 sensor and has an AoV of about 20^o, which is rather small. Obviously, the linear FoV and the AoV are important criteria in video sensor networks deployed for mission-critical surveillance applications. The DoV is a more subjective parameter. Technically, DoV could be very large but practically it is limited by the fact that an observed object must be sufficiently big to be identified.

2.1 Determining cover sets

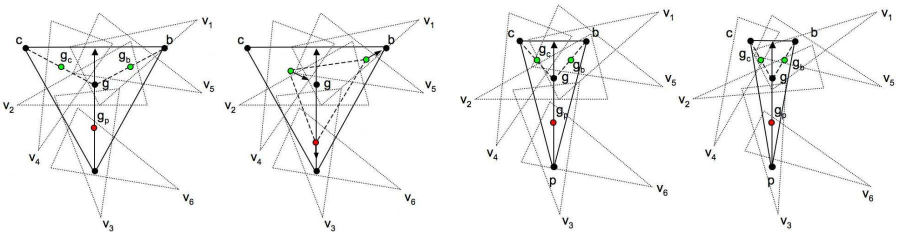
In the case of an omnidirectional sensing, a node can simply determine what parts of the coverage disc is covered by its neighbors. For the FoV coverage the task is more complex and determining whether a sensor's FoV is completely covered or not by a subset of neighbor sensors is a time consuming task which is usually too resource-consuming for autonomous sensors. A simple approach presented in [14] is to use significant points of a sensor's FoV to quickly determine cover sets that may not completely cover sensor v's FoV but a high percentage of it. First, sensor v can classify its neighbors into 3 categories of nodes, (*i*) those that cover point p, (*ii*) those cover point b and (*iii*) those that cover point c. Then, in order to avoid selecting neighbors that cover only a small portion of v's FoV, we add a fourth point taken near the center of v's FoV to construct a fourth set and require that candidate neighbors covers at least one of the 3 vertices and the fourth point. It is possible to use pbc's center of gravity, noted point g, as depicted in figure 1(a)(right). In this case, a node v can practically computes $Co(v)$ by finding the following sets, where $N(v)$ represents the set of neighbors of node v:

- $P/B/C/G = \{v' \in N(v) : v'$ covers point p/b/c/g of the FoV$\}$
- $PG = \{P \cap G\}$, $BG = \{B \cap G\}$, $CG = \{C \cap G\}$

Then, $Co(v)$ can be computed as the Cartesian product of sets PG, BG and CG ($\{PG \times BG \times CG\}$). However, compared to the basic approach described in [14], point g may not be the best choice in case of heterogeneous camera's AoV and very small AoV as will be explained in the next subsections.

2.2 The case of heterogeneous AoV

It is highly possible that video sensors with different angles of view are randomly deployed. In this case, a wide-angle FoV could be covered by narrow-angle FoV sensors and vice-versa. Figure 1(b) shows these cases and the left part of the figure shows the most problematic case when a wide FoV ($2\alpha = 60^o$) has to be covered by a narrow FoV ($2\alpha = 30^o$). As we can see, it becomes very difficult for a narrow angle node to cover *pbc*'s center of gravity g and one of the vertice at the same time.



(a) Heterogeneous AoV (b) Very small AoV

Fig. 2. Using more alternate points.

The solution we propose in this paper is to use alternate points g_p, g_b and g_c that are set in figure 2(a)(left) as the mid-point of segment [pg], [bg] and [cg] respectively. It is also possible to give different weights as shown in the right part of the figure. When using these additional points, it is possible to require that a sensor v_x either covers both c and g_c or g_c and g (the same for b and g_b, and p and g_p) depending on whether the edges or the center of sensor v's FoV are privileged. Generalizing this method by using different weights to set g_c, g_b and g_p closer or farther from there respective vertices can be useful to set which parts v's FoV has more priority as depicted in figure 2(a)(right) where g_c has moved closer to g, g_b closer to b and g_p closer to p.

2.3 The case of very small AoV

On some hardware, the AoV can be very small. This is the case for instance with the IMB400 multimedia board on the iMote2 which has an AoV of $2\alpha = 20^o$. Figure 2(b)(left) shows that in this case, the most difficult scenario is to be able to cover both point p and point g_p if g_p is set too far from p. As it is not interesting to move g_p closer to p with such a small AoV, the solution we propose is to discard point p and only consider point g_p that could move along segment [pg] as previously. Therefore in the scenario depicted in figure 2(b)(right), we have $PG = \{v_3, v_6\}$, $BG = \{v_1, v_2, v_5\}$ and $CG = \{v_4\}$ resulting in $Co(v) = \{\{v_3, v_1, v_4\}, \{v_3, v_2, v_4\}, \{v_3, v_5, v_4\}, \{v_6, v_1, v_4\}, \{v_6, v_2, v_4\}, \{v_6, v_5, v_4\}\}$.

2.4 Accuracy of the proposed method

Using specific points is of course approximative and a cover can satisfy the specific points coverage conditions without ensuring the coverage of the entire FoV.

To evaluate the accuracy of our cover sets construction technique, especially for very small AoV, we conducted a series of simulations based on the discrete event simulator OMNet++ (http://www.omnetpp.org/). The results were obtained from iterations with various node populations on a $75m.75m$ area. Nodes have random position P, random line of sight $\overrightarrow{V}$, equal communication ranges of $30m$ (which determines neighbor nodes), equal DoV of $25m$ and an offset angle α. We will test with $2\alpha = 20^o$ ($\alpha = \pi/18$), $2\alpha = 36^o$ ($\alpha = \pi/10$) and $2\alpha = 60^o$ ($\alpha = \pi/6$). We run each simulation 15 times to reduce the impact of randomness. The results (averaged over the 15 simulation runs) are summarized in table 1.

We will denote by CO_{pbcG}, $CO_{pbcApbc}$ and CO_{bcApbc} the following respective strategies: (i) the triangle points are used with g, which is pbc's center of gravity, when determining eligible neighbors to be included in a sensor's cover sets, (ii) alternates points g_p, g_b and g_c are used with the triangle points and, (iii) same as previously except that point p is discarded. The "stddev of %coverage" column is the standard deviation over all the simulation runs. A small standard deviation value means that the various cover sets have percentages of coverage of the initial FoV close to each other. When "stddev of %coverage" is 0, it means that each simulation run gives only 1 node with 1 cover set. This is usually the case when the strategy to construct cover sets is too restrictive.

Table 1 is divided in 3 parts. The first part shows the CO_{pbcG} strategy with $2\alpha = 60^o$, $2\alpha = 36^o$ and $2\alpha = 20^o$. We can see that using point g gives very high percentage of coverage but with $2\alpha = 36^o$ very few nodes do have cover sets compared to the case when $2\alpha = 60^o$. With very small AoV, the position of point g is not suitable as no cover sets are found. The second part of table 1 shows the $CO_{pbcApbc}$ strategy, where alternate points g_p, g_b and g_c are used along with the triangle vertices, with $2\alpha = 36^o$ and $2\alpha = 20^o$. For $2\alpha = 36^o$, this strategy succeeds in providing both a high percentage of coverage and a larger number of nodes with cover sets. When $2\alpha = 20^o$ the percentage of coverage is over 70% but once again very few nodes do have cover sets. This second part also shows CO_{bcApbc} (point p is discarded) with $2\alpha = 20^o$. We can see that this strategy is quite interesting as the number of nodes with cover sets increases for a percentage of coverage very close to the previous case. In addition, the mean number of cover sets per node greatly increases which is highly interesting as nodes with high number of cover sets could act as sentry nodes in the network. The last part of table 1 uses a mixed AoV scenario where 80% of nodes have an AoV of 20^o and 20% of nodes an AoV of 36^o. This last part shows the performance of the 3 strategies and we can see that CO_{bcApbc} presents the best tradeoff in terms of percentage of coverage, number of nodes with cover sets and mean number of cover sets per nodes when many nodes have a small AoV.

3 Criticality-based scheduling of randomly deployed nodes with cover sets

As said previously, the frame capture rate is an important parameter that defines the surveillance quality. In [12], we proposed to link a sensor's frame capture rate

Table 1. Results for CO_{pbcG}, $CO_{pbcApbc}$ and CO_{bcApbc}. $2\alpha = 20^o$, $2\alpha = 36^o$ and mixed AoV.

CO_{pbcG} 60^o #nodes	% nodes with coverset	mean % coverage	min,max % coverage/coverset	stddev of % coverage	min,max #coverset/node	mean #coverset/node
75	4.89	94.04	90.16,98.15	3.67	1,5.66	2.10
100	7.13	94.63	86.99,98.49	4.40	1,6	2.99
125	11.73	95.06	85.10,99.52	4.12	1,13	3.53
150	17.11	95.44	84,99.82	3.98	1,16.13	4.15
175	26.19	94.64	83.57,99.89	4.01	1,35.66	6.40

CO_{pbcG} 36^o #nodes	% nodes with coverset	mean % coverage	min,max % coverage/coverset	stddev of % coverage	min,max #coverset/node	mean #coverset/node
75	0	0	0,0	nan	0,0	0
100	1	92,03	89.78,98.64	0	1,1	1
125	1.87	91.45	88.83,93.15	2.97	1.13,2	1.56
150	1.78	95.06	91.47,98.19	4.06	1,3	1.94
175	3.43	94.42	87.60,99.03	4.40	1.13,2.66	1.92

CO_{pbcG} 20^o #nodes	% nodes with coverset	mean % coverage	min,max % coverage/coverset	stddev of % coverage	min,max #coverset/node	mean #coverset/node
all cases	0	0	0,0	nan	0,0	0

$CO_{pbcApbc}$ 36^o #nodes	% nodes with coverset	mean % coverage	min,max % coverage/coverset	stddev of % coverage	min,max #coverset/node	mean #coverset/node
75	12.44	77.48	56.46,91.81	13.13	1.13,9.13	3.62
100	20.13	79.62	53.65,98.98	12.05	1,10.66	3.94
125	30.67	76.89	50.53,97.92	11.58	1,34	5.40
150	35.11	78.47	52.07,96.09	10.60	1,31.13	6.90
175	48.57	77.76	49.97,98.10	10.54	1,50.13	11.57

$CO_{pbcApbc}$ 20^o #nodes	% nodes with coverset	mean % coverage	min,max % coverage/coverset	stddev of % coverage	min,max #coverset/node	mean #coverset/node
75	1.13	70.61	57.60,91.54	0	1,1	1
100	2	73.89	69.45,79.80	9.50	1.13,2	1.58
125	2.67	71.78	58.67,84.98	12.45	1.13,2	1.75
150	4	71.67	54.18,92.19	14.10	1,3.66	1.91
175	7.43	75.50	54.69,94.01	12.87	1,8	2.74

CO_{bcApbc} 20^o #nodes	% nodes with coverset	mean % coverage	min,max % coverage/coverset	stddev of % coverage	min,max #coverset/node	mean #coverset/node
75	7.56	73.79	56.18,88.54	12.45	1,5	2.10
100	9.13	67.16	47.78,88.71	13.80	1,4.66	2.14
125	12.53	70.12	40.41,87.46	13.11	1,11.13	3.17
150	21.13	70.10	45.72,91.57	11.57	1,19.13	4.18
175	25.13	71.79	44.15,94.18	11.91	1,37	7.05

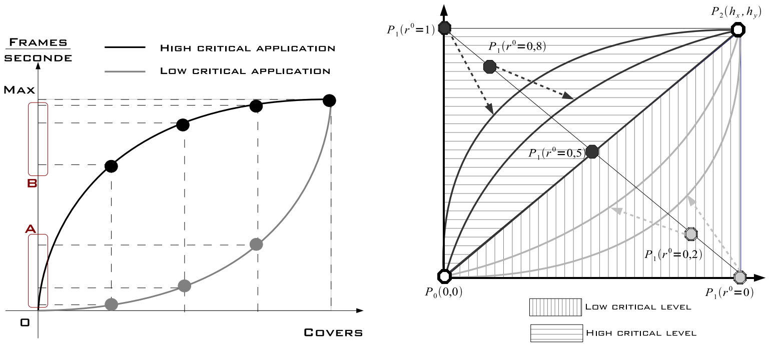
CO_{pbcG} 20^o(80%) 36^o(20%) #nodes	% nodes with coverset	mean % coverage	min,max % coverage/coverset	stddev of % coverage	min,max #coverset/node	mean #coverset/node
75,100,125	0	0	0,0	nan	0,0	0
150	0.66	92.13	83.64,95.83	0	1,1	1
175	0.57	93.45	85.75,96.14	0	1,1	1

$CO_{pbcApbc}$ 20^o(80%) 36^o(20%) #nodes	% nodes with coverset	mean % coverage	min,max % coverage/coverset	stddev of % coverage	min,max #coverset/node	mean #coverset/node
75	3.11	81.89	78.13,89.02	8.15	1.13,2	1.58
100	3	69.83	65.50,74.55	8.18	1,3.66	1.89
125	4.80	78.58	69.52,90.92	8.03	1,3.13	1.56
150	8.67	78.12	56.41,97.59	13.71	1,5	1.95
175	10.19	76.60	50.4,95.47	13.48	1,8.66	2.62

CO_{bcApbc} 20^o(80%) 36^o(20%) #nodes	% nodes with coverset	mean % coverage	min,max % coverage/coverset	stddev of % coverage	min,max #coverset/node	mean #coverset/node
75	9.13	81.48	69.18,93.72	9.72	1,5.66	2.06
100	6	80.10	62.82,90.16	11.81	1,3.66	1.94
125	10.93	73.15	47.14,92.14	14.43	1.13,9.13	3.65
150	20	72.12	45.53,95.94	12.19	1,16.66	4.83
175	20.95	75.15	43.01,97.57	12.59	1,18.13	5.15

to the size of its cover set. In our approach we define two classes of applications: low and high criticality applications. This criticality level can oscillate from a concave to a convex shape as illustrated in Figure 3 with the following interesting properties:

- **Class 1 "low criticality"**, does not need high frame capture rate. This characteristic can be represented by a concave curve (figure 3(a) box A), most projections of x values are gathered close to 0.
- **Class 2 "high criticality"**, needs high frame capture rate. This characteristic can be represented by a convex curve (figure 3(a) box B), most projections of x values are gathered close to the max frame capture rate.



(a) Application classes (b) The Behavior curve functions

Fig. 3. Modeling criticality.

[12] proposes to use a Bezier curve to model the 2 application classes. The advantage of using Bezier curves is that with only three points we can easily define a ready-to-use convex (high criticality) or concave (low criticality) curve. In figure 3(b) $P_0(0,0)$ 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 cover cardinality and h_y is the maximum frame capture rate determined by the sensor node hardware capabilities. As illustrated in Figure 3(b), by moving the behavior point P_1 inside the rectangle defined by P_0 and P_2, we are able to adjust the curvature of the Bezier curve, therefore adjusting the risk level r^0 introduced in the introduction of this paper. According to this level, we define the risk function called Rk which operates on the behavior point P_1 to control the BV function curvature. According to the position of point P_1 the Bezier curve will morph between a convex and a concave form. As illustrated in figure 3(b) the first and the last points delimit the curve frame. This frame is a rectangle and is defined by the source point $P_0(0,0)$ and the threshold point $P_2(h_x, h_y)$. The middle point $P_1(b_x, b_y)$ defines the risk level. We assume that this point can move through the second diagonal of the defined rectangle $b_x = \frac{-h_y}{h_x} * b_y + h_y$. Table 2 shows the corresponding capture rate for some relevant values of r^0. The cover set cardinality $|Co(v)| \in [1, 12]$ and the maximum frame capture rate is set to 3fps.

Table 2. Capture rate in fps when P2 is at (12,3).

r^0	1	2	3	4	5	6	7	8	9	10	11	12
0	.01	.02	.05	0.1	.17	.16	.18	.54	.75	1.1	1.5	3
.1	.07	.15	.15	.17	.51	.67	.86	1.1	1.4	1.7	2.1	3
.4	.17	.15	.55	.75	.97	1.1	1.4	1.7	2.0	2.1	2.6	3
.6	.16	.69	1.0	1.1	1.5	1.8	2.0	2.1	2.4	2.6	2.8	3
.8	.75	1.1	1.6	1.9	2.1	2.1	2.5	2.6	2.7	2.8	2.9	3
1	1.5	1.9	2.1	2.4	2.6	2.7	2.8	2.9	2.9	2.9	2	3

4 Fast event detection with criticality management

We are evaluating in this section the performance of an intrusion detection system by investigating the stealth time of the system. For these set of simulations, 150 sensor nodes are randomly deployed in a $75m * 75m$ area. Unless specified, sensors have an 36^o AoV and the $CO_{pbcApbc}$ strategy is used to construct cover sets. Each sensor node captures with a given number of frames per second (between 0.01fps and 3fps) according to the model defined in figure 3(b). Nodes with 12 or more cover sets will capture at the maximum speed. Simulation ends when there are no active nodes anymore.

4.1 Static criticality-based scheduling

We ran simulations for 4 levels of criticality: $r^0 = 0.1$, 0.4, 0.6 and 0.8. The corresponding capture rates are those shown in table 2. Nodes with high capture rate will use more battery power until they run out of battery (initial battery level is 100 units and 1 captured image consumes 1 unit) but, according to the scheduling model, nodes with high capture rate are also those with large number of cover sets. Note that it is the number of valid cover sets that defines the capture rate and not the number of cover sets found at the beginning of the cover sets construction procedure. In order to show the benefit of the adaptive behavior, we computed the mean capture rate for each criticality level and then used that value as a fixed capture rate for all the sensor nodes in the simulation model. $r^0 = 0.1$ gives a mean capture rate of 0.12fps, $r^0 = 0.4$ gives 0.56fps, $r^0 = 0.6$ gives 0.83fps and $r^0 = 0.8$ gives 1.18fps. Table 3 shows the network lifetime for the various criticality and frame capture rate values.

Table 3. Network lifetime.

$r^0 = 0.1$	0.12 fps	$r^0 = 0.4$	0.56 fps	$r^0 = 0.6$	0.83 fps	$r^0 = 0.8$	1.18 fps
2900s	620s	1160s	360s	560s	240s	270s	170s

Using the adaptive frame rate is very efficient as the network lifetime is $2900s$ for $r^0 = 0.1$ while the 0.12fps fixed capture rate last only $620s$. In order to evaluate further the quality of surveillance we show in figure 4(top) the mean stealth time when $r^0 = 0.1$, $fps = 0.12$, $r^0 = 0.4$ and $fps = 0.56$, and in figure

4(bottom) the case when $r^0 = 0.6$, $fps = 0.83$, $r^0 = 0.8$ and $fps = 1.18$. The stealth time is the time during which an intruder can travel in the field without being seen. The first intrusion starts at time 10s at a random position in the field. The scan line mobility model is then used with a constant velocity of 5m/s to make the intruder moving to the right part of the field. When the intruder is seen for the first time by a sensor, the stealth time is recorded and the mean stealth time computed. Then a new intrusion appears at another random position. This process is repeated until the simulation ends.

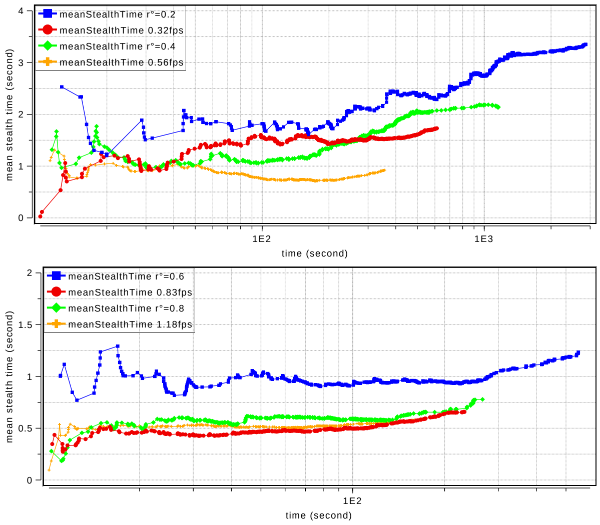


Fig. 4. Mean stealth time. Top: $r^0 = 0.1$, $fps = 0.12$, $r^0 = 0.4$, $fps = 0.56$. Bottom: $r^0 = 0.6$, $fps = 0.83$, $r^0 = 0.8$, $fps = 1.18$.

Figure 5(left) shows for a criticality level $r^0 = 0.6$ the special case of small AoV sensor nodes. When $2\alpha = 20^o$, we compare the stealth time under the $CO_{pbcGpbc}$ and the CO_{bcGpbc} strategies. Discarding point p in the cover set construction procedure gives a larger number of nodes with larger number of cover sets, as shown previously in table 1. In figure 5(left) we can see that the stealth time is very close to the $CO_{pbcGpbc}$ case while the network lifetime almost doubles to reach 420s instead of 212s. The explanation is as follows: as more nodes have cover sets, they act as sentry nodes allowing the other nodes to be in sleep mode while ensuring a high responsiveness of the network.

In addition, for the particular case of disambiguation, we introduce a 8m.4m rectangle at random positions in the field. $CO_{pbcGpbc}$ is used and $2\alpha = 36^o$. The rectangle has 8 significant points as depicted in figure 5(right) and moves at the velocity of 5m/s in a scan line mobility model (left to right). Each time a sensor node covers at least 1 significant point or when the rectangle reaches the right boundary of the field, it appears at another random position. This process

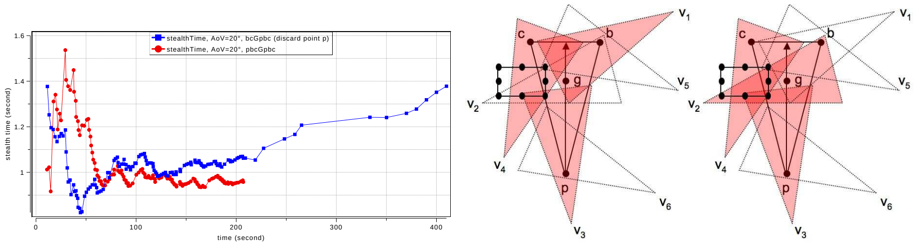


Fig. 5. Left: Stealth time, sliding winavg with 20 samples batch, $r^0 = 0.6$, AoV=20^o, $CO_{pbcGpbc}$ and CO_{bcGpbc}. Right: Rectangle with 8 significant points, initial sensor v and 2 different cover sets.

starts at time $t = 10s$ and is repeated until the simulation ends. The purpose is to determine how many significant points are covered by the initial sensor v and how many can be covered by using one of v's cover set. For instance, figure 5(right) shows a scenario where v's FoV covers 3 points, the left cover set ($\{v_3, v_1, v_4\}$) covers 5 points while the right cover set ($\{v_3, v_2, v_4\}$) covers 6 points. In the simulations, each time a sensor v covers at least 1 significant point of the intrusion rectangle, it determines how many significant points are covered by each of its cover sets. The minimum and the maximum number of significant points covered by v's cover sets are recorded along with the number of significant points v was able to cover initially. Figure 6 shows these results using a sliding window averaging filter with a batch window of 10 samples. We can see that node's cover sets always succeed in identifying more significant points. Figure 7 shows that with the rectangle intrusion (that could represent a group of intruders instead of a single intruder) the stealth time can be further reduced.

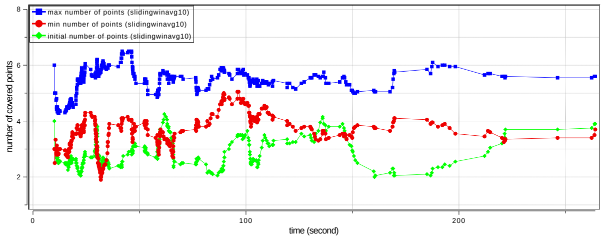


Fig. 6. Number of covered points of an intrusion rectangle. Sliding winavg of 10.

4.2 Dynamic criticality-based scheduling

In this section we are presenting preliminary results in dynamically varying the criticality level during the network lifetime. The purpose is to only set the surveillance network in an alerted mode (high criticality value) when needed, i.e.

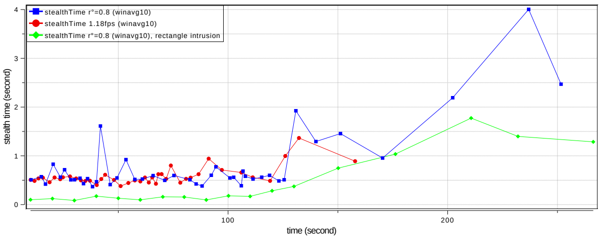


Fig. 7. Stealth time, winavg with 10 samples batch, $r^0 = 0.8$, $fps = 1.18$ and $r^0 = 0.8$ with rectangle intrusion.

on intrusions. With the same network topology than the previous simulations, we set the initial criticality level of all the sensor nodes to $r^0 = 0.1$. As shown in the previous simulations, some nodes with large number of cover sets will act as sentries in the surveillance network. When a sensor node detects an intrusion, it sends an alert message to its neighbors and increases its criticality level to $r^0 = 0.8$. Alerted nodes will then also increase their criticality level to $r^0 = 0.8$. Both the node that detects the intrusion and the alerted nodes will run at a high criticality level for an alerted period, noted T_a, before going back to $r^0 = 0.1$. Nodes may be alerted several times but an already alerted nodes will not increase its T_a value any further in this simple scenario. As said previously, we do not attempt here to optimize the T_a value nor using several level of criticality values. Figure 8shows the mean stealth time with this dynamic behavior. T_a is varied from 5s to 60s. We can see that this simple dynamic scenario already succeeds in reducing the mean stealth time while increasing the network lifetime when compared to a static scenario that provides the same level of service.

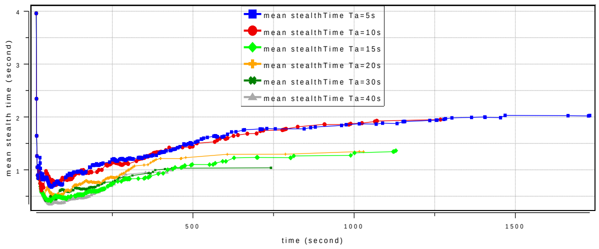


Fig. 8. Mean stealth time with dynamic criticality management.

5 Conclusions

This paper presented the performances of cover sets construction strategies and dynamic criticality scheduling that enable fast event detection for mission-critical surveillance with video sensors. We focused on taking into account cameras with heterogeneous angle of view and those with very small angle of view. We show that our approach improves the network lifetime while providing low stealth time in case of intrusion detection systems. Preliminary results with dynamic criticality management also show that the network lifetime can further be increased. These results show that besides providing a model for translating a subjective criticality level into a quantitative parameter, our approach for video sensor nodes also optimize the resource usage by dynamically adjusting the provided service level.

Acknowledgment. This work is partially supported by the FEDER POCTEFA EFA35/08 PIREGRID project, the Aquitaine-Aragon OMNI-DATA project and by the PHC Tassili project 09MDU784.

References

1. H. F. R. Collins, A. Lipton and T. Kanade: Algorithms for cooperative multisensor surveillance. In: Proceedings of the IEEE, vol. 89, no. 10 (2001).
2. T. Yan, T. He, and J. A. Stankovi: Differentiated surveillance for sensor networks. In: ACM SenSys (2003).
3. T. He et al.: Energy-efficient surveillance system using wireless sensor networks. In: ACM MobiSys (2004).
4. S. Oh, P. Chen, M. Manzo, and S. Sastry: Instrumenting wireless sensor networks for real-time surveillance. In: International Conference on Robotics and Automation (2006).
5. R. Cucchiara et al.: Using a wireless sensor network to enhance video surveillance. In: J. Ubiquitous Computing and Intelligence, vol. 1, no. 2, (2007).
6. L. N. I. Stoianov and S. Madden: Pipenet: A wireless sensor network for pipeline monitoring. In: ACM IPSN (2007).
7. M. Albano and R. D. Pietro: A model with applications for data survivability in critical infrastructures. In: J. of Information Assurance and Security, vol. 4 (2009).
8. O. Dousse, C. Tavoularis, and P. Thiran: Delay of intrusion detection in wireless sensor networks. In: ACM MobiHoc (2006).
9. Y. Zhu and L. M. Ni: Probabilistic approach to provisioning guaranteed qos for distributed event detection. In: IEEE INFOCOM (2008).
10. E. Freitas et al.: Evaluation of coordination strategies for heterogeneous sensor networks aiming at surveillance applications. In: IEEE Sensors (2009).
11. M. Keally, G. Zhou, and G. Xing: Watchdog: Confident event detection in heterogeneous sensor networks. In: IEEE Real-Time and Embedded Technology and Applications Symposium (2010).
12. A. Makhoul, R. Saadi, and C. Pham: Risk management in intrusion detection applications with wireless video sensor networks. In: IEEE WCNC (2010).
13. M. Rahimi et al.: Cyclops: In situ image sensing and interpretation in wireless sensor networks. In: ACM SenSys (2005).
14. A. Makhoul and C. Pham: Dynamic scheduling of cover-sets in randomly deployed wireless video sensor networks for surveillance applications. In IFIP Wireless Days (2009).