C O P C A M S

Cognitive & Perceptive Cameras

Artemis-JU GA n°332913

D5.4
Large Area Surveillance Applications Report

WP5 –Applications & Field Tests

Editors:
Hüseyin Özkan, Metin Aktaş, Önder Altan (ASEL),
Christian Fabre (CEA).

Date: 2017-01-17 at 15:19

Version: v2.0

Status: Public version

CEA ref. number: LETI/DACLE/16-0855

http://copcams.eu

Document History

Date	Author	Modification
2016-06-21	Metin Aktaş (ASEL) Önder ALTAN	First Draft
2016-07-13	Jordi Serra (CTTC) David Pubill, David López Ignacio Llamas, Nikos Bartzoudis Christos Verikoukis	CTTC's contribution
2016-07-13	Fabio Poiesi (QMUL)	QMUL's contribution (section 2.2.1)
2016-07-14	Cyril BERGERON (TCS)	TCS's contribution (section 2.2.6)
2016-09-08	Önder ALTAN (ASEL)	Editorial checked
2016-09-22	Christian Fabre (CEA)	Added CEA contrib.
2016-09-23	Julie Foucault (CEA)	Added CEA ref. number and updated status
2016-09-23	Joaquim Llinares (CCTL) Arantxa Montalban Juan Luis de Amaya Robles Juan Leandro Sánchez	Added CCTL contrib.
2016-12-05	Juan Leandro Sánchez (CCTL)	Updated CCTL's contrib.
2016-12-05	Christian Fabre (CEA)	Updated QMUL's contrib.

COPCAMS Partners

CEA	Commissariat à l'énergie atomique et aux énergies alternatives
TCS	THALES Communications & Security SA
TRT-FR	THALES Research & Technology France (repr. THALES SA)
INRIA	Institut national de recherche en informatique et automatique
CTTC	Centre Tecnolòlogic de Telecomunicacions de Catalunya
CCTL	Concatel
IQU	Iquadrat Informatica S.L.
TECN	Tecnalia Research & Innovation
TED	Tedesys Global S.L.
UC	Universidad de Cantabria
GUT	Politechnika Gdańska
JSI	Institut ``Jožef Stefan''
DTU	Danmarks Tekniske Universitet / IMM
~~TRT-UK~~	~~THALES Research & Technology (UK) Ltd~~
QMUL	Queen Mary University of London
ASEL	ASELSAN Electronics Industry
KTOR	Kolektor Group d.o.o.
SOG	Sogilis
SQST	Squadrone system
TUKL	Thales UK Limited

Table of Contents

List of Figures

List of Tables

1 Introduction

This document reports the demonstration activities to show the effectiveness of COPCAMS solutions for the large area surveillance applications. The demonstration activities was performed in two categories, i.e., a field test and laboratory experiments.

The aim of Large Area Surveillance Application field test is to effectively monitor large areas with multiple cameras and extract meaningful information about the monitored area such as locating and classifying the moving object(s). The moving objects in the monitoring area are classified as 'human', 'vehicle' or 'other' based on either only the view captured by the camera at the central station or all the views available, i.e., views of the end node cameras in addition to the one at the central station.

2 Demonstration Performance Results

2.1 Field Test

This section describes the field test scenario for "Large Area Surveillance Application" that is based on the requirements given in "D1.1 & D1.2 – Summary of Functional & Non-Functional Description" and use cases defined in "D1.4 – Summary of Use Cases and Field Test Definition" documents. The evaluation strategy and measurements to be collected are also described in this section.

ASELSAN developed algorithms regarding the detection and classification of objects in an area of interest that is monitored by single camera. In this section, we first summarize the proposed approach for the field test and define test scenario applied for performance evaluations. Then we define the performance evaluation metric that are measured in the test scenario. We finalize this section by given the measured test results.

2.1.1 Proposed Approach

In the field test of Large Area Surveillance Application, we concentrate on an approach with the following properties:

- Objects are not tracked, instead; the video stream is split into several spatio-temporal volumes named "cell"s of a pre-specified size, i.e., m x n x T, where m: cell row size, n: cell column size and T: memory. Here, the cells can be chosen overlapping or non-overlapping. In the preferred implementation, we opt to have the cells to overlap in order to sufficiently cover a target activity.
- At every t'th frame during the data stream, each cell declares a result of that
    - An activity exists
        - Activity corresponds to a "human" /"vehicle",

        - Activity corresponds to a rare/abnormal activity: "other"
    - An activity does not exist.
- Unsupervised: the algorithm expects no classification input from the user to facilitate the task. However, a supervised framework can be also used, in case of which; the developed algorithm would be slightly modified.
- No correlation model among cells is assumed or learned, i.e., they are processed independently. Hence, the cell processing is strongly parallelizable.
- Two phases: training and operation. In phase of the training, the algorithm is fed with data and let learn its parameters.

Based on this approach, we develop and propose an algorithm, which detects an object and classifies it as "human", "vehicle", and "other", where the class of "other" is defined as any abnormal activity. The proposed algorithm consists of the phases training and operation. Before we explain the details of our algorithm, we point out that in both of these phases, the video is transformed into a feature space. In the following, we explain the details of this transformation.

Feature Extraction: For a given video stream V(1:M, 1:N, 1:T) whether in the training phase or operational phase, where M is the frame row size, N is the frame column size and T is the length of the stream (possibly infinite),

- For each frame, the image gradients Ix, Iy and motion flow Jx, Jy are first calculated.
- Then at each t'th frame V(1:M,1:N,t) and for each cell, i.e., for all cells $v(k_1, k_2)=V(k_1i+1: k_1i+m, k_2j+1: k_2j+n,t)$'s with k_1 and k_2 spanning all possible cells, where i and j are determining the overlapping amounts of these cells in the row and column, respectively,
    - For every moving pixel in $v(k_1, k_2)$, i.e., a pixel that has a sufficiently large motion magnitude, the descriptor a=[x,y,t,|Ix|,|Iy|,sqrt(Ix^2+Iy^2),|Jx|,|Jy|, sqrt(Jx^2+Jy^2)] is computed, where (x,y,t) is the coordinate of that pixel with respect to the centroid of the moving blob in that cell.
    - Based on this computed descriptor, a feature vector (61 dimensional)

      $f(k_1, k_2, t)$ = [corr(a), var(a), imgrad_hist, optflow_hist] (all vectorized) is computed. Here, corr(a) is the normalized correlation matrix of the descriptor within the cell, var(a) is the corresponding variance, imgrad_hist is the undirected histogram vector of the image gradients within the cell and similarly; optflow_hist is the undirected histogram vector of the optical flow vectors within the cell. Here, we use 8-bin undirected histograms.

Training Phase: Let T1 (finite) be the length of a provided training video, then once all the features vectors are computed for each cell,

- A training data set of f(k1, k2,t)'s for all k1, k2,1<t<T1 is obtained as the collection of the feature vectors obtained from all cells across the training video.
- Then, an anomaly detection algorithm is applied to extract the nominal/typical activities.
- The nominal activities are then clustered into a sufficiently large number of clusters.

Extracting the Nominal Statistics in the Training Phase: Let S = {x: x = f(k1, k2,t) for all k1, k2,1<t<T1} be the training set of data obtained from a given T1-length training video which does only include the nominal activities, i.e., only human and vehicle observations can exists in S. Then x follows an unknown nominal density x~pX, whereas the class of "other" is assumed to be drawn from a uniform density sharing the same support with pX. Based on this, we formulate the detection of the "other" objects as an anomaly detection problem under these density assumptions such that when given a test instance, the anomaly detection (detection of "other") probability is maximized with a pre-specified constant false alarm rate. To this end, using the training data S, the sufficient statistics regarding the nominal density pX are extracted.

To be more precise, a score value l(x) is assigned to every training instance x in S, which is defined as $l(x) = \frac{1}{|S|} \sum_{\forall y \in S: y \neq x} d_K(x) \leq d_K(y)$, where $d_\kappa(x)$ is the K nearest neighbor of x in S. Here, the score value l(x) can be regarded as the popularity measure estimating the corresponding density level of x on pX. Since we assume that the anomalous objects follow a uniform density, the detection rate is known to be maximized when the $(1-\tau)$ rate of the nominal activity is covered in a minimum volume set, where τ is the desired constant false alarm rate. This minimum volume set is precisely estimated in the asymptotically consistent sense by the introduced score values l(x) [1]. For an example, if a test instance x is declared as anomalous/other whenever $l(x) \leq \tau$, the detection probability of the anomalous/other objects is maximized with a constant false alarm rate τ In the Figure 1 (taken from [1]), we illustrate the output of this anomaly detection approach on a simple 2-dimensional data set.

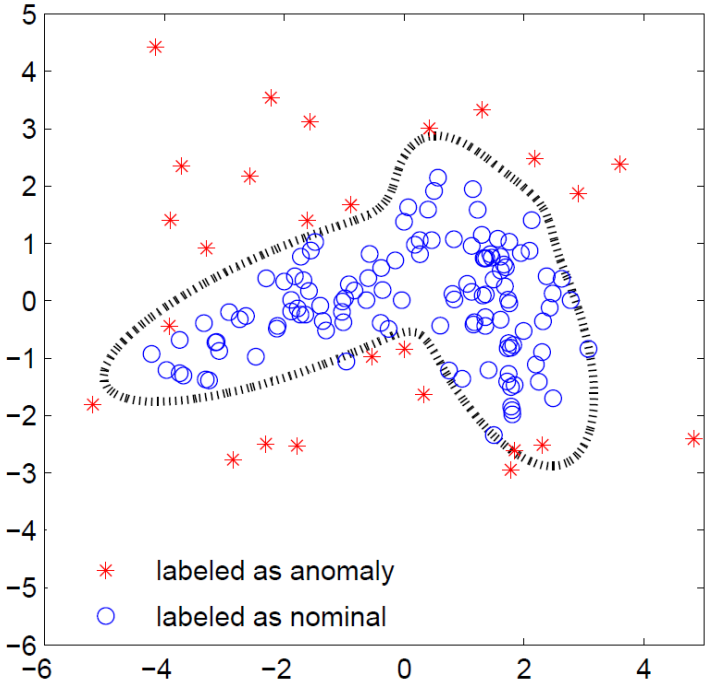


Figure 1: The output of the anomaly detection approach on a simple 2-dimensional data set [1].

Clustering the Nominal Activity in the Training Phase: We note that when the described anomaly detection algorithm is applied on the training data, the typical/nominal activities, consisting of human and vehicle observations in the context of COPCAMS, are obtained. In order to separate these human and vehicle observations from each other and further discover the sub-classes, we propose a novel clustering algorithm based on the score levels that are defined and used in the anomaly detection approach. Since the proposed clustering algorithm similar to the well known mean-shift approach in spirit, we name it as the "score shift" algorithm. However, in contrast to the mean-shift algorithm, the proposed score shift clustering generates clusters with feasible cluster centers. Namely, the cluster centers are real observations in a sense that they cannot be a mixture of, for instance, a vehicle and human observation as in the case of mean-shift. This allows us to assign an instance the label of the corresponding cluster center as vehicle or human. This would not be possible with the mean shift. Moreover, the phase of clustering and equivalently classification of the data is computationally only dependent on the outputs of the anomaly detection approach that are computed a priori, which makes it extremely fast and efficient with no or very little further computations.

Score Shift Clustering in the Training Phase: Note that for every training instance x in S, we have a score value l(x). Then the proposed score shift algorithm works as follows: Starting from x, a walk is designed such that the first step of this walk is from x to its nearest neighbor y in S such that $l(y)>l(x)$. Then we take the second step from y to z, and the third step from z to another and so on. This procedure is guaranteed to converge an element in S, which is then assigned as a representative for x. Repeating this procedure for every instance x, we immediately obtain a clustering of the training data, with feasible

cluster centers. In Figure 2, we illustrate an example of score shift clustering. In this example, we project the described 61 dimensional observations into the principle eigenvectors corresponding to the largest two eigenvalues.

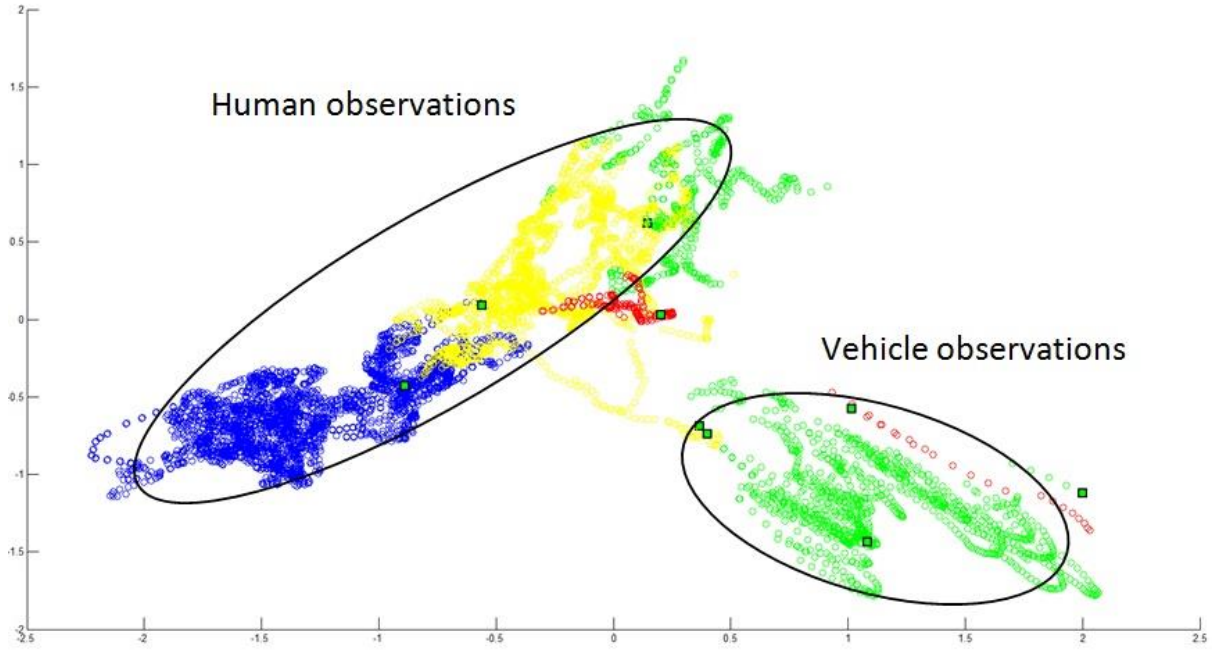


Figure 2: An example illustration of score shift clustering.

As shown, the proposed clustering method is able to discriminate the two nominal activity of human and vehicle with several other sub-clusters.

Operational Phase: In the operational phase, again, the feature vector defined above is computed for each cell. Then for a cell, this feature is tested whether it is a nominal activity. If it is found to be not-nominal, then "other" is declared. If it is found to be nominal, then it is clustered with respect to the determined clusters in the training phase. Then, if the found cluster includes a human activity "human" is declared; otherwise, "vehicle" is declared.

Remark: We emphasize that a manual inspection of the training results is required to determine which cluster corresponds to human and which corresponds to vehicle, which is far more practical compared to the supervised methods. Moreover, this explicit labeling might not even be necessary.

Remark: Using the standard Euclidean distance between correlation matrices in clustering the training data might not be suitable. In this case, a specific distance transformation can be utilized.

2.1.2 Test Scenario

The field test is performed in a test area in ASELSAN's facility with the test setup illustrated in Figure 3. In this setup, there are two fixed wide field of view (FOV) cameras and one narrow FOV Pan

Tilt Zoom (PTZ) camera located on the surface of the building in the test area. The setup parameters of the cameras are given in Table 1. In demonstration, Samsung SNP-3120VH camera is used, whose specifications are listed in Table 1.

We test our single camera solution on a scene, where the usual activities are automobiles are entering/ leaving the scene with varying speeds as well as casually walking pedestrians. The proposed method has been trained on six different sequences including only the usual activities and tested on three sequences, which additionally include a vehicle (to simulate an anomalous, i.e., ``other" type of an activity) that is a van.

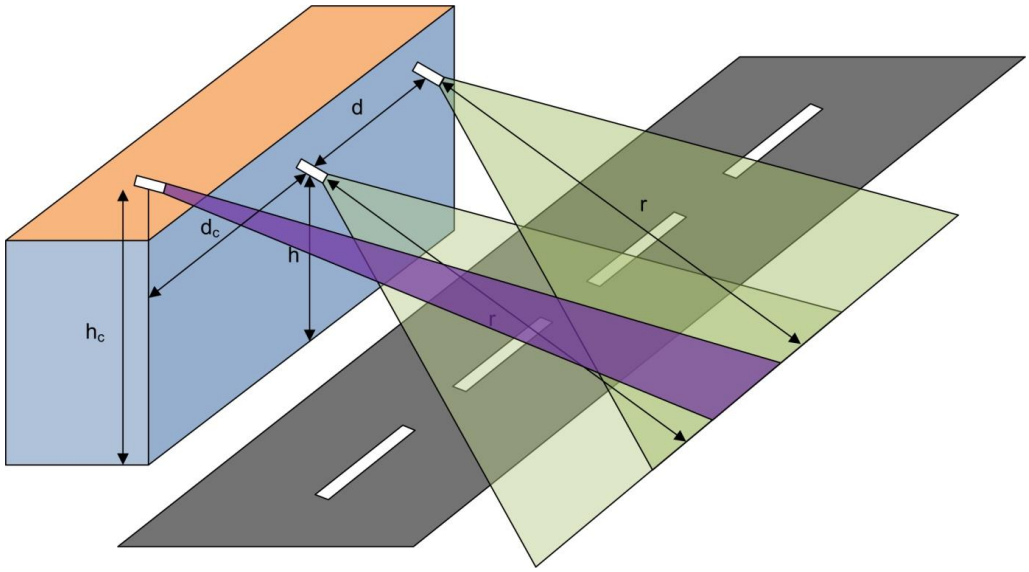


Figure 3: The illustration of test setup for Large Area Surveillance Application filed test

Table 1: The setup parameters of the cameras that are illustrated in Figure 3.

Camera Type	FOV	Relative Positions			Rotation	
		X	Y	Z	Azimuth	Elevation
Fixed	Horizontal: 54.44˚ Vertical: 42.32˚	10	0	7	0	-10
PTZ	Horizontal: 4.62˚ Vertical: 3.58˚	0	0	0	0	-10

Table 2: The specifications of Samsung SNP-3120VH camera

Camera Parameter		**Value**
Video	Imaging Device	1/4" Ex-view HAD PS CCD
	Total Pixels	NT : 811(H) x 508(V), PAL : 795(H) x 596(V)
	Effective Pixels	NT : 768(H) x 494(V), PAL : 752(H) x 582(V)
	Scanning System	Progressive(VPS ON) (If WDR on, Interlaced Scan)
	Frequency	NT : H : 15.734KHz / V : 59.94Hz, PAL : H : 15.625KHz / V : 50Hz
	Horizontal Resolution	Color : 600 TV lines
	Min. Illumination	Color : 0.7 Lux (F 1.65, 50 IRE, VPS OFF), 0.001 Lux (Sens up 512X) B/W : 0.07 Lux (F 1.65, 50 IRE, VPS OFF), 0.0001 Lux (Sens up 512X)
	S / N Ratio	50dB
	Video Out	CVBS : 1.0 Vp-p / 75Ω composite
Lens	Focal Length (Zoom Ratio)	3.69~44.32mm (12X)
	Max. Aperture Ratio	F1.65(Wide) / F2.01(Tele)
	Angular Field of View	H : 54.44˚(Wide) ~ 4.62˚(Tele) / V : 42.32˚(Wide) ~ 3.58˚(Tele)
	Min. Object Distance	0.2m (Wide) / 0.8m (Tele)
	Lens Type	DC Auto Iris
Pan / Tilt / Rotate	Pan Range	360˚ Endless
	Pan Speed	Preset : 650˚/sec, Manual : 0.05˚/sec ~120˚/sec (Turbo:200˚/sec)
	Tilt Range	-5˚~185˚
	Tilt Speed	Preset : 650˚/sec, Manual : 0.05˚/sec ~120˚/sec
Network	Ethernet	RJ-45 (10/100BASE-T)
	Video Compression Format	H.264, MPEG4, MJPEG

	Resolution	NT : 704x480, 640x480, 352x240, 320x240 PAL : 704x576, 640x480, 352x288, 320x240
	Max. Framerate	NT : 30fps / PAL : 25fps
	Video Quality Adjustment	H.264/MPEG4 : Compression Level, Target Bitrate Level Control MJPEG : Quality Level Control
	Bitrate Control Method	H.264/MPEG4 : CBR or VBR MJPEG : VBR
	Streaming Capability	Multiple Streaming (Up to 10 Profiles)
	IP	IPv4, IPv6
	Protocol	TCP/IP, UDP/IP, RTP(UDP), RTP(TCP), RTSP, NTP, HTTP, HTTPS, SSL, DHCP, PPPoE, FTP, SMTP, ICMP, IGMP, SNMPv1/v2c/v3(MIB-2), ARP, DNS, DDNS
	Security	HTTPS(SSL) Login Authentication Digest Login Authentication IP Address Filtering User access Log
	Streaming Method	Unicast / Multicast
	Max. User Access	10 users at Unicast Mode
	Web Viewer	Supported OS : Windows XP / VISTA / 7, MAC OS Supported Browser : Internet Explorer 6.0 or Higher, Firefox, Google Chrome, Apple Safari
	Central Management	Software NET-i viewer

2.1.3 Evaluation & Measurements

The described scenario in Section 2.1.2 page 11 takes place outdoor under 'sufficient' and 'stable' day light or artificial illumination with enough lux percentage, in addition to that no background change is allowed to happen, e.g., an object of the background such as a parked bicycle is assumed to stay in its place with no motion throughout the scenario. We assume that the system is correctly installed to cover the monitoring area and the cameras are calibrated.

Under these conditions, the following metrics are measured for the performance evaluation.

- **Detection Rate:** For an evaluation of the first step in the single camera solution, i.e., the "other" / "anomaly" detection step, the detection rate of anomalies is measured at a given range of bearable false alarm rates.
- **False Alarm Rate:** For an evaluation of the first step in the single camera solution, i.e., the "other" / "anomaly" detection step, the false alarm rate of anomalies is measured at a given range of desired detection rates.
- **Classification Accuracy:** In the set of observations that are labeled as "nominal", i.e., "not anomalous" or "other", the accuracy for the task of "human" vs "vehicle" classification is measured.
- **Total Transmission Bandwidth:** For an evaluation of communication overhead in distributed surveillance architecture, the total transmission bandwidth for a unit time or unit event is measured.

2.1.4 Results

We train the proposed single camera algorithm on 6 video sequences and test on 3 different test sequences (several snapshots of which can be found below) with the following system parameters, where the actual values that we use in our implementation are stated within parentheses.

- vt, (M,N,Zf): The proposed algorithm operates on an input video sequence vt, whose frames are M × N **(576x704)** and Zf **(739)** is the length of the sequence that can be possibly infinity.
- (Mb,Nb,Zb): We process the input vt block by block, where each block is a video chunk whose frames are Mb ×Nb (144x176) and Zb (10) is the length of the block sequence that is finite. The blocks are placed on a rectangular grid on the imaging plane.
- (βM, βN): The amount of overlaps between blocks. The right/left neighbor of a block is βMMb/βNNb (βM=βN=0.5) pixels away on the right/left direction.
- τv, τvc: τv (0.5) is the threshold for determining the significant motion magnitudes both in declaring activity for a pixel and creating the motion histogram. τvc (5) is the threshold for consistency for a block activity in time.

- τI : This is the threshold (20) to determine the significant image gradients in calculating the histogram of gradients.
- va, vb, vac: vb (0.5) is the border threshold for determining the center of a block that scales the spatial dimensions of a block. va (0.03) is the frame level activity threshold for a block and vac (0.5) is the activity consistency threshold for a block in time.
- K: This is the size of the neighborhood (10 x log10 x total number of features).
- τFA : This is the false alarm rate in declaring anomalies (0.005).

Note that we skip the details of the parameter selection of the Lucas Kanade (LK) optical flow algorithm, since it is not a focus of this study. In fact, any other algorithm extracting the motion field can replace the LK algorithm.

In the first and second test sequences, we simulate an anomaly as a van passing through the scene which is an ``abnormally" large vehicle compared to the other vehicles included in the training sequences; except that in the third test sequence, there is no anomalous activity. Hence, in the third sequence, we only have a human vs vehicle classification task.

Recall that for every block, we first declare an anomaly decision as anomalous vs nominal; and if found anomalous, we declare a second decision stating the classification of the observed activity. Therefore, in order to measure the anomaly detection rate, we compute the rate of anomaly decisions for the blocks extracted from the van (the anomalous vehicle in the first and second test sequences). As for the false alarm rate, we compute the rate of anomalous decisions for the blocks extracted from the other activities (not from the van).

In order to measure the classification accuracy, for every nominal block that is also truly labeled as nominal, we compute the rate of accuracy for the declared human vs vehicle classification.

The following table summarizes our findings in the test sequences.

Table 3: The test results

	Detection Rate	**False Alarm Rate**	**Classification Accuracy**
Test Sequence-1	65 %	0 %	94 %
Test Sequence-2	98 %	4 %	100 %
Test Sequence-3	--	--	100 %

As for the total transmission bandwidth in a distributed processing scenario, we emphasize that each agent (node or processing unit) should communicate the 61 dimensional feature vectors with each other, where a feature vector is computed for each block. Therefore, for instance in the test sequence-2 that includes a relatively crowded scene with multiple vehicles and humans, since we process approximately 0.1 block per frame, the total transmission burden is approximately calculated as (if each feature is ``double" requiring 64 bits in the worst case; not that less number of bits can also be used to encode each feature) 0.1x61x64= 390 bits per frame, which roughly correspond to only 48 ``int8" pixels per frame. Here, if each feature is encoded using 16 bits (which is sufficient for most of the practical cases), the required bandwidth turns out as low as 12 pixels per frame.

In the following we present several snapshots obtained during the operational phase:

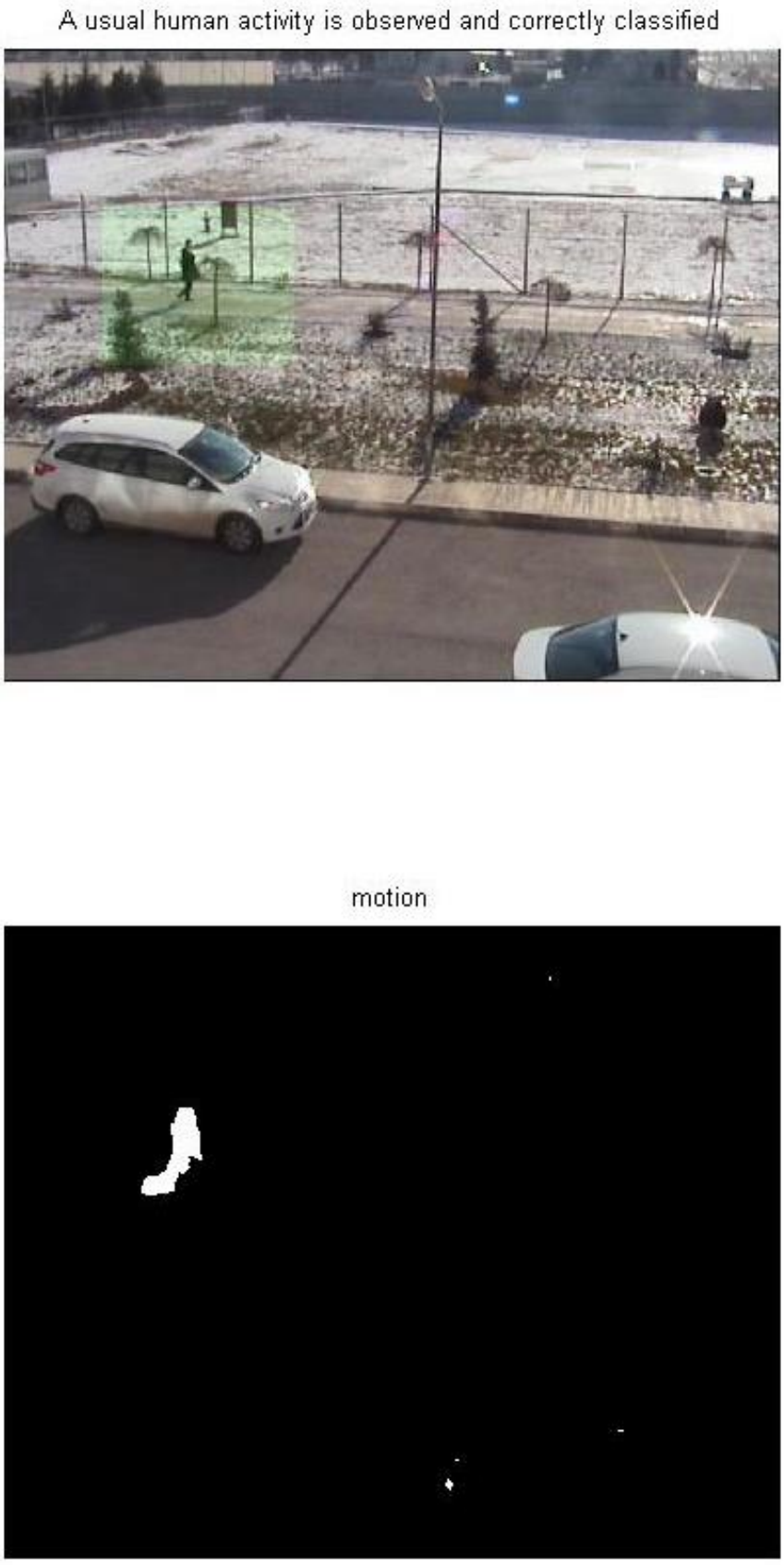


Figure 4: A usual human activity observation

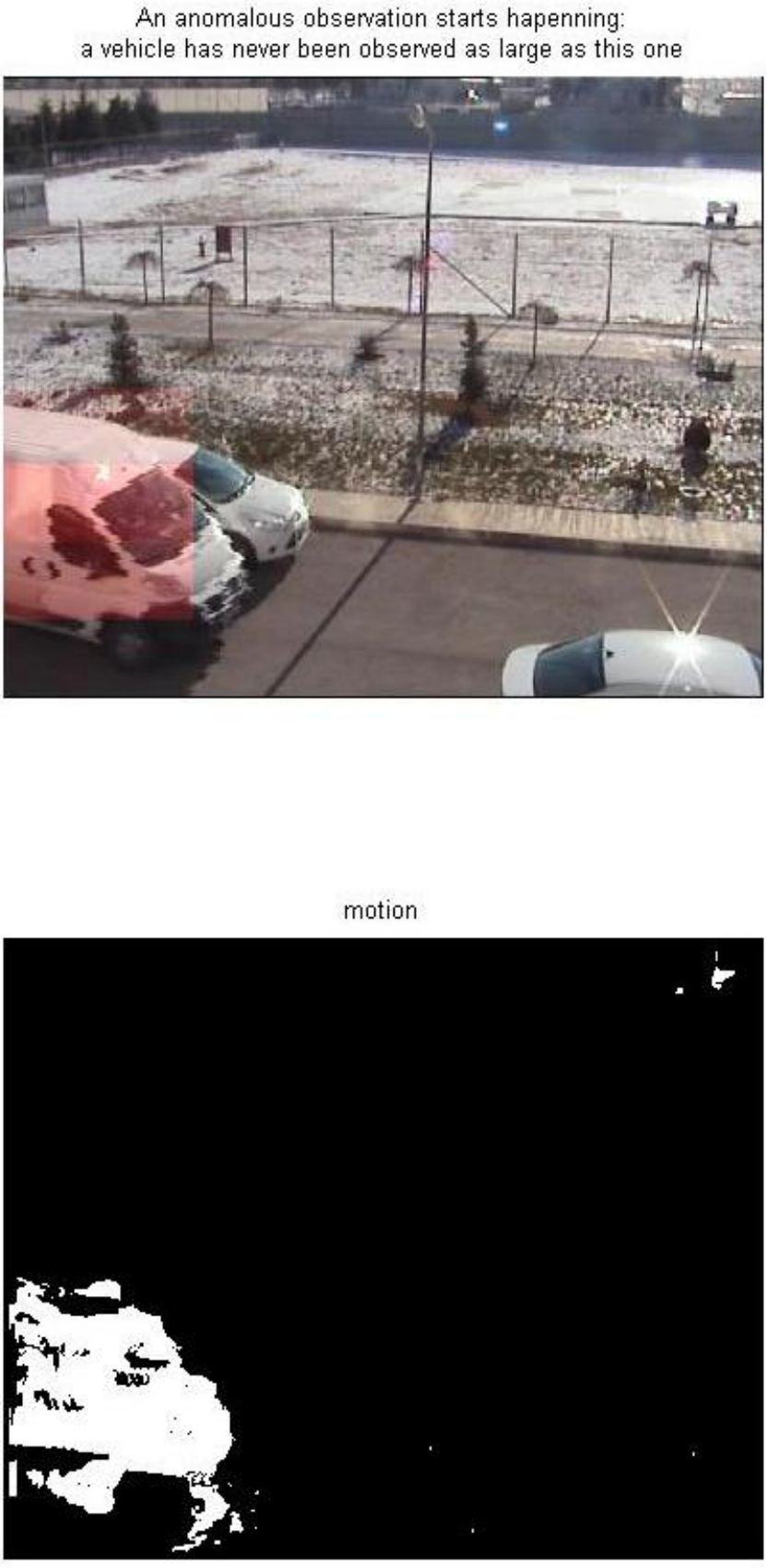


Figure 5: An anomalous activity observation

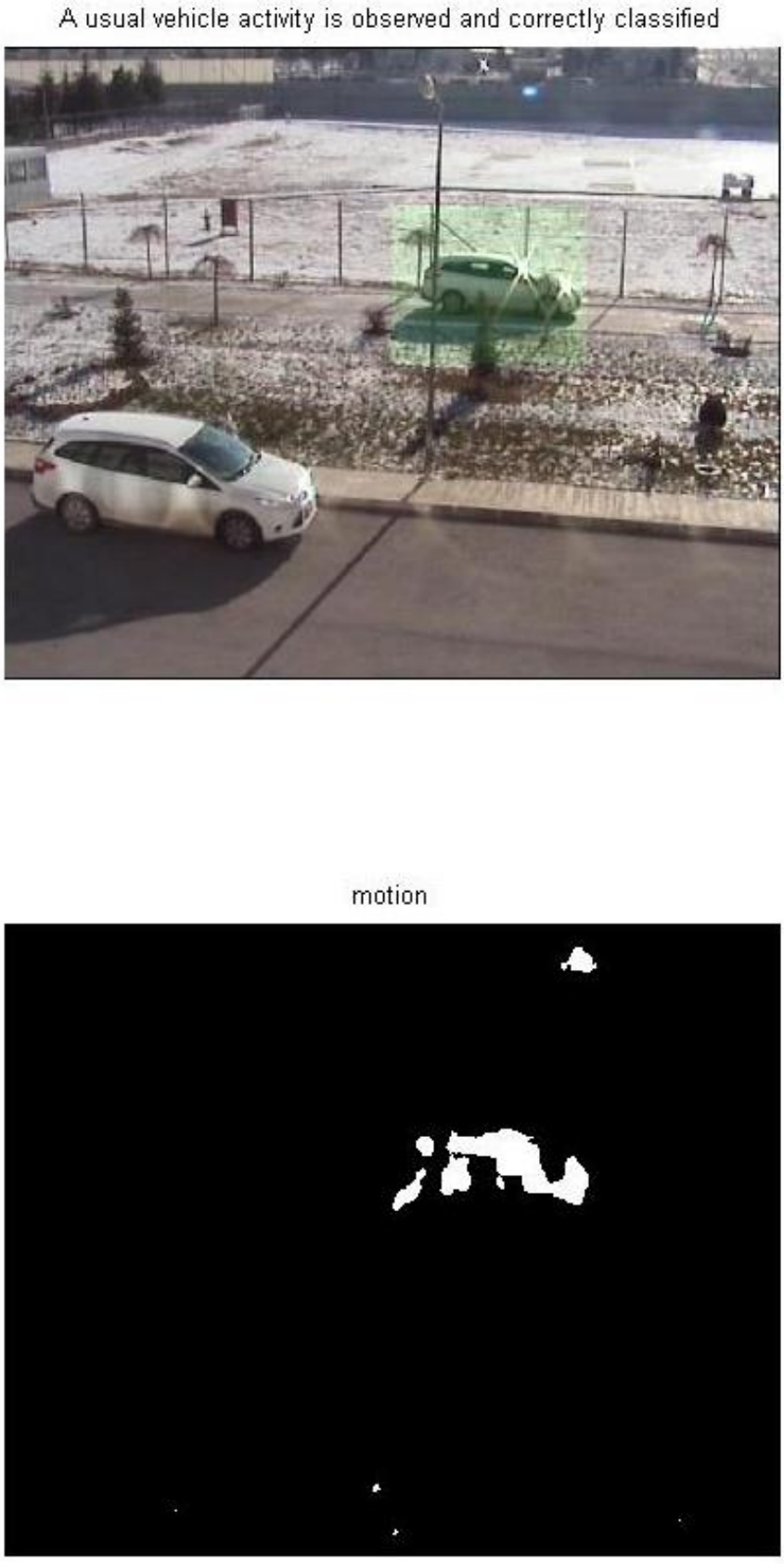


Figure 6: A usual vehicle activity observation.

2.2 Lab Experiments

2.2.1 Multi-target Detection and Tracking Experimental Setup

2.2.1.1 Motivation

The aim of this work is to parallelize the Probability Hypothesis Density Particle Filter (PHD-PF) [29] on NVIDIA Jetson TK1 to achieve a runtime performance of 15 frames-per-second (fps) for a set of given detections.

PHD-PF is a probabilistic tracker that estimates the state of the targets (e.g. position, velocity) using a posterior probability distribution. The posterior distribution is computed via the Bayes rule that employs a prior distribution (prediction) and a likelihood function (update). Each mode of this distribution models a likely target state. PHD-PF approximates the posterior distribution using Monte Carlo sampling with a finite number of samples (particles). This approximation uses a set of particle states and weights. The prediction is often calculated using an affine transformation and additive Gaussian noise to the particle states. The likelihood function computes the similarity among states and measurements (detections). The main difference among standard multi-target particle filters and PHD-PF is that the latter can predict the target cardinality over time, so that it is more robust in situations of noisy measurements. The prediction of the target cardinality is formulated as an additional calculation before the weight update.

In a nutshell, the PHD-PF target state estimation and tracking involves the following steps in a recursive way: i) particle state and weight prediction from *k-1* to *k*, ii) weight update using detections, iii) resampling, iv) state estimation via particle clustering and v) association of estimated states at *k* with states at *k-1* to build trajectories. Because particle clustering may under- and/or over-estimate clusters, we use ad-hoc methods for cluster merge and split to refine the state estimation. Clustering is performed using Expectation-Maximization (E-M) as it can be initialized using the expected number of clusters and initial cluster states.

2.2.1.2 Scenario definition

We test and validate the performance of PHD-PF on NVIDIA Jetson TK1 using PETS2009-S2L1 dataset (http://goo.gl/UNCCCl). The scene contains a minimum of three person targets and a maximum of six. Examples of tracking results are shown in Figure 7 and Figure 8. We use

a person detector based on Histogram of Oriented Gradients (HOG) [30] that is provided in OpenCV in both CPU and GPU version.

Figure 7: Tracking example obtained with HOG detector and PHD-PF on PETS2009-S2L1.

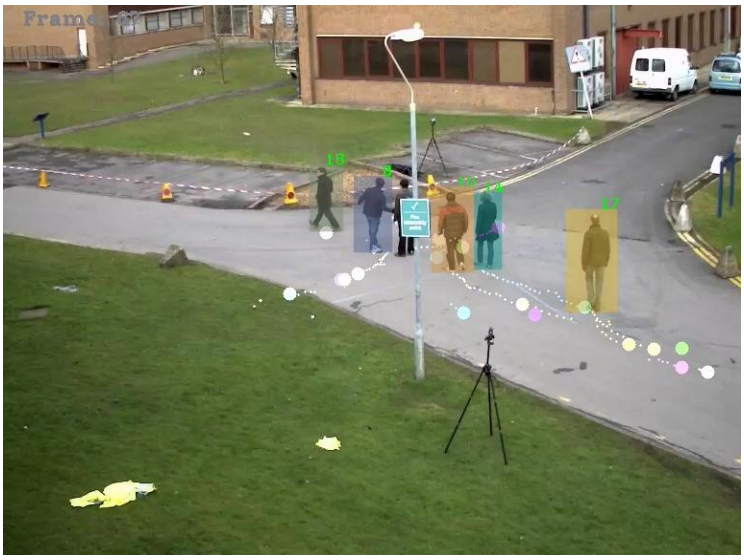


Figure 8: Tracking example obtained with HOG detector and PHD-PF on PETS2009-S2L1.

2.2.1.3 Parallelization procedure

Profiling was performed on the PHD-PF using PETS2009-S2L1. The most computationally expensive function is found to be the E-M clustering. We opted for a GPU implementation and, because our PHD-PF is based on OpenCV libraries that do not offer the GPU implementation of E-M, we adapted a Matlab function for E-M based on CUDA libraries [31] to our NVIDIA Jetson TK1.

We then analyzed the code thoroughly and identified three hotspots that can be parallelized. We also tried the parallelization of other parts of the code but without success due to intrinsic dependencies of data. The parallelization is only possible when operations computed on a certain data set are independent.

During parallelization it is important to evaluate whether the overhead cost of data management to be distributed among different cores is greater or lesser than the actual computation cost if there were computed on a single core. Via experiments we observed that the overhead cost for these three functions plus their computation was less than the actual computation on a single core. Therefore, for these three hotspots, we used OpenMP libraries and distributed the computation among the 4 cores of Jetson CPU (ARM Cortex-A15). OpenMP uses the following *pragma* syntax to perform the parallelization

```
#pragma omp parallel for num_threads(4)
```

We observed that the specification of the number of threads is an important step. This is a way to force Jetson to use all its 4 cores. Otherwise the compiler would need to guess or interrogate the system in order to retrieve the number of available cores, and if some of the cores are disabled for power saving reasons, the parallelization could be performed for less core (e.g. 2).

2.2.1.4 Evaluation strategy and measurements

We evaluate the runtime performance of the PHD-PF with and without parallelization by averaging the results over 150 frames on PETS2009-S2L1. We show the results of five versions of the implementation namely: CPU (the original version), GPU only (with E-M CUDA version), GPU + OpenMP 1 (with OpenMP applied to weight update), GPU + OpenMP 2 (with OpenMP applied to weight update and split cluster) and GPU + OpenMP 3 (with OpenMp applied to weight update, split cluster and resampling). We evaluate the performance of the tracker using the following accuracy measures: Multi-Object Tracking Accuracy (MOTA), Multi-Object Tracking Precision (MOTP), Precision, Recall and Identity Switches (IDS) [32].

As we can see from the results in Figure 9, the performance dramatically improves when the E-M clustering is implemented on GPU. The improvement goes from about 17 times, in case with 300 particles per target, to about 23 times in the case with 1300 particles per target. When

the CPU parallelization with OpenMP is applied the performance increase even more getting to about 28 times faster in the former case and about 48 times faster in the latter case. It is interesting to observe that the overhead cost tends to influence the performance lesser by increasing the computation load given to both CPU and GPU. The performance is very close to our goal set at the beginning of the parallelization process (15fps) as we achieved about 14fps for the case with 300 particles. Table 4 shows the tracking accuracy at varying number of particles per target.

2.2.1.5 Future developments

We are working towards the real-time implementation of a full detection and tracking pipeline on an embedded platform (e.g. NVIDIA Jetson TK1). Our ultimate goal is to be able to achieve 15fps including the detection stage.

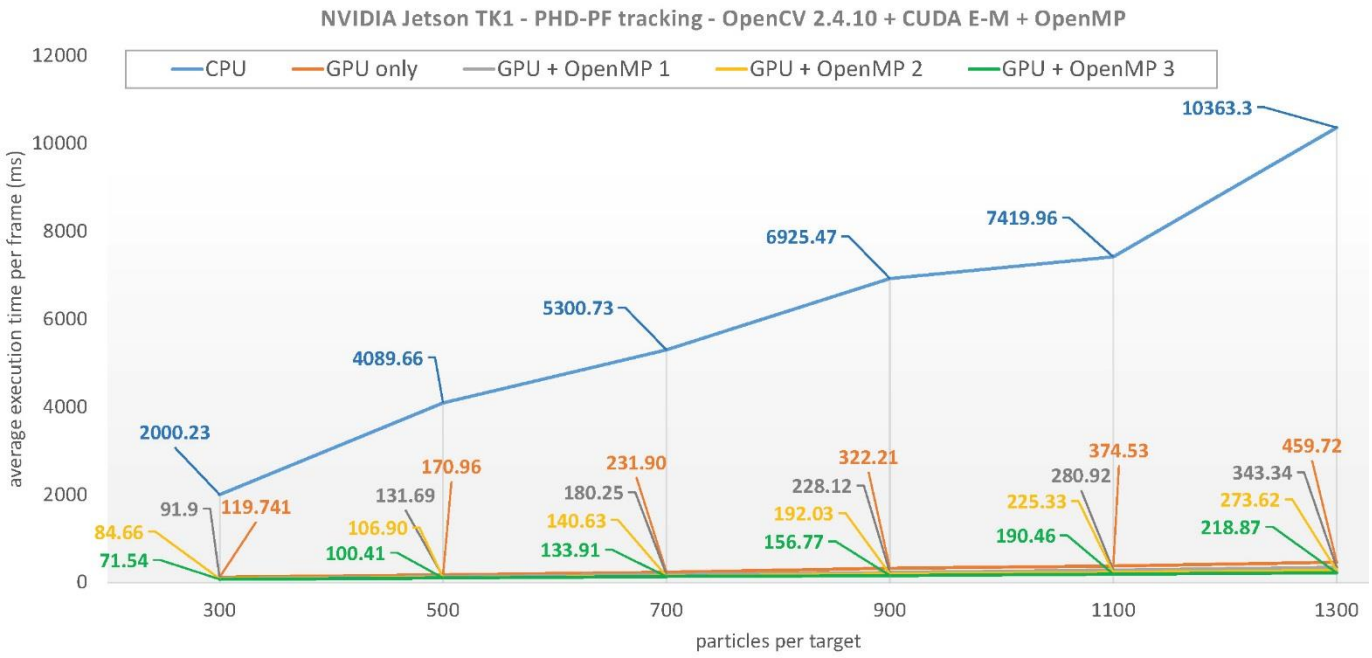


Figure 9: Runtime performance (ms) with varying number of particles per target.

Table 4: Tracking accuracy results with varying number of particles per target.

# particles/target	MOTA	MOTP	Precision	Recall	IDS

300	3.9	62.0	56.9	39.0	101
500	3.7	61.5	56.5	38.5	93
700	5.7	61.5	58.2	39.7	98
900	7.5	61.7	58.6	41.2	83
1100	10.3	61.6	60.9	41.2	80
1300	11.4	61.9	60.7	43.3	70

2.2.2 Distributed Detection of Events Based on WSN For Large Area Surveillance

2.2.2.1 Introduction

In a previous deliverable, D5.1, CTTC proposed a system relying on a WSN, whose aim is to carry out a distributed detection of security events. The aim, of the contribution to this deliverable D5.4, is to assess the performance of the proposed distributed event detection system. To this end, the objective is to assess the detection metrics, i.e. detection and false alarm rate, as a function of the system parameters and event parameters that have a direct influence on them. Moreover, the evaluation strategy described in D5.1 is taken into account to carry out the system performance evaluation. Namely, a twofold strategy is envisaged. First, the system will be evaluated in a simulated scenario. This permits to vary some parameters that have a direct influence on the performance, e.g. the amplitude of the event. This simulated scenario is fundamental to obtain a deep insight of the influence of the event and system parameters into the performance of the proposed detection system. The second scenario is a real scenario deployed in a lab environment, where the conclusions extracted from the simulated scenario must be validated.

2.2.2.2 Simulated scenario

In this section we present the simulator that was implemented to assess the performance of the proposed distributed detection system. Moreover, we extract conclusions related to the performance assessment thanks to a campaign of numerical simulations.

2.2.2.2.1 Description of the simulator

We begin this section by recalling the proposed distributed detection system proposed in D5.1, as the simulator must emulate its behavior. In Figure 10 this system is displayed. It can be observed that it consists of a set of sensors which monitor an area under control and whose aim is to detect events that occur within this area. Each sensor is connected to a Zolertia Z1 unit, which has processing and communication capabilities. The pair sensor and Zolertia Z1 unit form what is called a WSN node in the sequel. Each WSN node takes a decision about the presence or absence of an event in a batch processing basis. That is, it considers a buffer of sensor measurements and then takes a detection decision. This procedure is repeated for each new buffer of sensor measurements. The detection decisions that the WSN nodes take at each buffer are sent to the fusion centre via a wireless single hop link, by means of an IEEE 802.15.4 radio interface. The fusion centre is composed of a Z1 unit, which acts as a sink that permits to receive IEEE 802.15.4 packets and a Raspberry Pi 2. The Z1 unit is connected to the Raspberry Pi 2 via USB, whose aim is to gather the decisions of each WSN node and implement a fusion of detections algorithm which improves the individual decisions of the WSN nodes. Furthermore, the Raspberry Pi 2 module paves the way to communicate with other COPCAMS modules thanks to its USB, Wi-Fi and Ethernet ports.

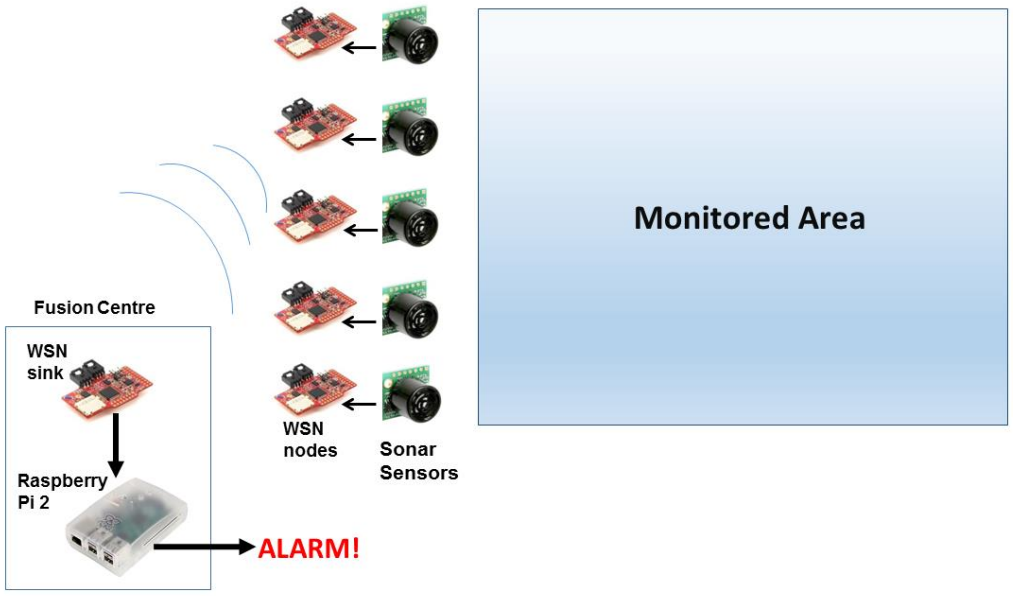


Figure 10: Setup of the detection system based on sonar sensors.

The simulator that pretends to model the event detection system exposed in Figure 10 consists of the next functional blocks:

1) Generation of the sensor measurements at each processing buffer.
2) Event Detection at each WSN node.
3) Computation of performance metrics for each WSN node.
4) Fusion of detections algorithm.
5) Computation of performance metrics after fusion algorithm.
6) Plot of performance metrics.

The functionalities implemented by those blocks depend on a set of system, event and performance metrics parameters, which are listed in Table 5, Table 6 and Table 7, respectively. Next we proceed to explain in detail the functional blocks as well as the role of the parameters which model their behavior.

Table 5: System parameters

$\boldsymbol{\sigma^2}$	Noise power at each sensor.
M	Number of sensors.
k	Parameter k of the k out of M fusion algorithm.

h_i	Attenuation factor of event at the *i*-th sensor.
hf	Heuristic factor of the detection threshold at each sensor.
N	Length of the processing buffer.
B	Total number of processing buffers.
d_{min}	Minimum distance that sensors can measure.
P_{event}	Probability that an event is present at a given processing buffer.

Table 6: Event parameters

L	Length of the event (at a given buffer) in samples.
A	Amplitude of the event (in cm). This is the amplitude that the sensors should measure if there was not attenuation.

Table 7: Performance metrics

D_{si}	Detection rate at *i*-th sensor.
F_{si}	False alarm rate at *i*-th sensor.
D_f	Detection rate after fusion algorithm.
F_f	False alarm rate after fusion algorithm.

1) Generation of the sensor measurements at each processing buffer.

The first task of the simulator is to generate the data measurements of each sensor which contain the noise plus event signals and which feed the detection algorithms at each WSN node. To this end, first a matrix $W \in \mathbb{R}^{Mx(NB)}$ is generated containing only noise. Each element of Wis assumed to be independent and identically distributed (in a statistical sense) and it is generated according to a Gaussian distribution with mean d_{min} and variance σ^2. That is, the *i-th* row of this matrix contains all the samples of the experiment related to the *i-th* sensor, whose mean is the minimum distance that the sensor can measure and with a given noise power, to be specified below.

Next, we have to generate the events. To this end, we assume that the presence or absence of an event at a given processing buffer is random. Namely, it is assumed that the presence of an event at a given processing buffer follows a binomial distribution, whose parameters are the probability that an event is present P_{event} and the probability than an event is not present $1 - P_{event}$.

Regarding the *n-th* samples of the event signal $x(n)$, it is assumed that it can be modeled as a rectangular waveform of length L, denoted by $\Pi_L(n)$, and amplitude A

$$x(n) = A\Pi_L(n).$$

The event measured at each sensor undergoes an attenuation, denoted by h_i for the *i-th* sensor. This attenuation is due to the wireless channel and the beam pattern associated to the sensors. Thereby, the event signal measured at the *n-th* sample of the *i-th* sensor can be written as

$$s_i(n) = h_i x(n) = h_i A\Pi_L(n).$$

Therefore, according to the above definitions, the samples measured at a given processing buffer of the *i-th* sensor are modeled as

$$y_i(n) = \beta s_i(n) + w_i(n)$$

where $\beta = 1$ with probability P_{event}, i.e. it models the presence or absence of the event at the current buffer. Moreover, according to the above explanations, $w_i(n) \sim N(d_{min}, \sigma^2)$ is the noise.

2) Event Detection at each WSN node.

Given the signal measured at a given processing buffer, the task of the event detection algorithm is to decide between the hypothesis that an event is present, denoted by H_1, and the hypothesis that the event is not present, denoted by H_0. According to the above definitions, these hypotheses can be expressed as follows at a given WSN node *i*,

$$H_0: y_i(n) = w_i(n)$$

$$H_1: y_i(n) = s_i(n) + w_i(n) = h_i A\Pi_L(n) + w_i(n)$$

Next, the detector considered to discern between these two hypotheses is explained.

Heuristic threshold detector

This type of detection algorithm is based on setting a heuristic detection threshold above the noise. That is, it decides H_1 if any of the buffer samples is above the threshold. In the sim-

ulator, the floor level is estimated during a training period at the beginning of the first processing buffer. Namely, the mean $\mu = d_{min}$ and the variance of the noise σ^2 are estimated by means of their sample estimators $\hat{\mu}$ and $\hat{\sigma}^2$,

$$\hat{\mu} = \hat{d}_{min} = \sum_{m=1}^{T} y_i(m)$$

$$\hat{\sigma}^2 = \sum_{m=1}^{T} (y_i(m) - \hat{\mu})^2$$

where *T* is the number of training samples. Thereby, the threshold, γ_h, for the heuristic detector is given by,

$$\gamma_h = \hat{\mu} + \hat{\sigma} + hf$$

Thereby, the detection algorithm is based on the next methodology,

$$If\ y_i(n) > \gamma_h\ Decide\ H_1, otherwise\ decide\ H_0\ .$$

3) Computation of performance metrics for each WSN node.

In order to assess the performance of the detection methods, at each node, some metrics must be computed. In our context, these performance metrics are the detection rate and the false alarm rate. The detection rate is denoted by D and it is defined as

$$D = \frac{Number\ of\ decisions\ event\ is\ present\ (when\ effectively\ event\ is\ present)}{Number\ of\ cases\ when\ event\ is\ present}.$$

Regarding the false alarm rate, which is denoted by *F*, it is defined as

$$F = \frac{Number\ of\ decisions\ event\ is\ present\ (when\ event\ is\ actually\ not\ present)}{Number\ of\ cases\ when\ event\ is\ not\ present}$$

4) Fusion of detections algorithm.

This functionality in the real system is implemented in the fusion centre. This is a method that receives as inputs the detection decisions taken at each WSN node, for a given processing buffer, and it provides a single detection decision which must improve the individual decisions of each node.

The fusion algorithm considered herein is based on the k out of M fusion rule, which is a widely used methodology, see e.g. [21]. This method considers the M decisions taken by the M WSN nodes at a given processing buffer, where the decision of the *i-th* sensor is denoted by u_i. It is assumed that $u_i = 1$ means that the *i-th* WSN node detects an event, whereas $u_i = -1$ means that the *i-th* WSN node does not detect an event. The global decision, i.e. the decision at the output of the fusion rule, is denoted by u and the fusion rule decides $u = 1$ (i.e. that an event is present) if

$$u_1 + \cdots + u_M \geq 2k - M$$

whereas $u = -1$ otherwise. This means that to decide that an event is present at least *k* WSN nodes must detect the event. Note that *k* is a user parameter and that logical AND and OR functions are a special case of the *k* out of *M* fusion rule. In the sequel, it is considered that *k*=1, that is an OR logical function is considered, which means that if one sensor detects an event, then the global decision decides that an event is present.

5) Computation of performance metrics after fusion algorithm.

The detection performance must be assessed again after the implementation of the fusion rule, described in the previous point. The aim is to corroborate that the performance is improved thanks to the fusion. Thereby, as in the case of the individual WSN nodes, the performance metrics considered for the fusion algorithm are the detection rate and the false alarm rate. These are defined as above, though they are applied after the fusion rule.

6) Plot of performance metrics.

This functionality of the simulator aims to plot the performance metrics, i.e. detection and false alarm rates, corresponding to each WSN node and after the fusion rule. This permits to assess the performance of the proposed system as a function of the event and system parameters. More insights will be given in the next section.

2.2.2.2.2 Numerical results

In this section we pretend to carry out a campaign of simulation results based on the simulator exposed in the previous sections. The aim is to assess the performance of the distributed detection system as a function of the event and system parameters. Unless otherwise stated, in the next simulations the event length is set to L=5, the processing buffer length to N=10, the total number of processing buffers B=10000 and the probability of an event at a given processing buffer $P_{event} = 0.3$, $d_{min} = 15$ cm

First we assess the influence of the event amplitude when the noise variance is small. This is shown in Figure 11, where plots of detection rate and false alarm rate are displayed for a fixed noise standard deviation of $\sigma = 2$ and where the event amplitude is fixed to A=10 cm in the left plot and A=40 cm in the right plot. Moreover, in these figures the heuristic factor *hf*, which determines the threshold of the heuristic detector, is varied from 0 (which corresponds to a threshold at the noise floor) and 52 which is a threshold above the event signal. 0 shows that when the noise variance is small, there is an interval of *hf* values where almost perfect detection performance can be achieved (i.e. D≈1 and F≈0). This *hf* interval corresponds to values above the noise level and below the event amplitude. Moreover, Figure 11 shows that as the event amplitude increases, the *hf* margin for perfect detection increases as well.

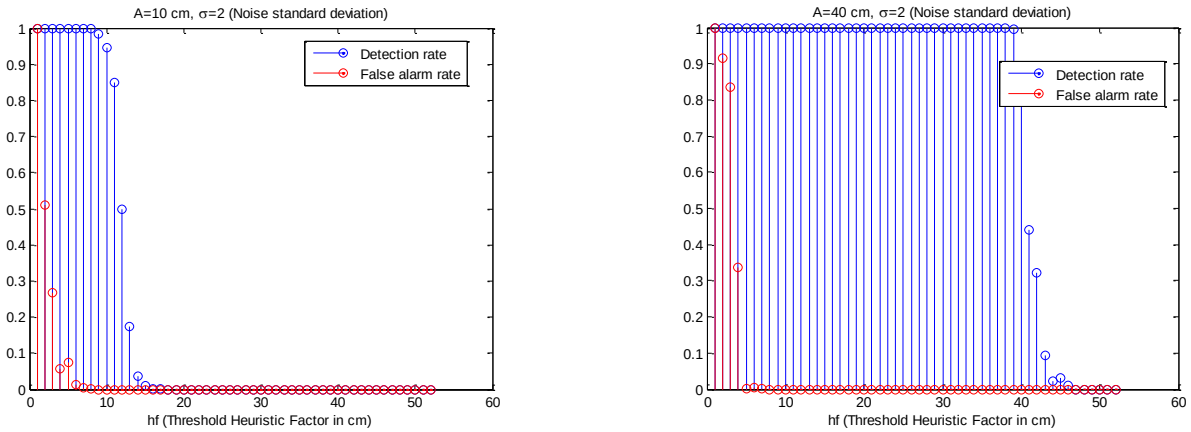


Figure 11: Influence of the event amplitude given a small noise variance.

In Figure 12 the influence of increasing the noise variance, for a given event amplitude, is assessed. On the left hand a small noise standard deviation ($\sigma = 2$) is presented, whereas in the right hind side a noise standard deviation $\sigma = 15$, which is a significant level compared to the event signal, is presented. Figure 12 highlights that increasing the noise power tends to corrupt more the sensor measurements and thereby it is more difficult to detect the presence of the event. This is translated in an increment of the false alarm rate and a reduction of the detection

rate in the interval of 0 < hf < A, where before for a small noise power an almost perfect detection could be achieved.

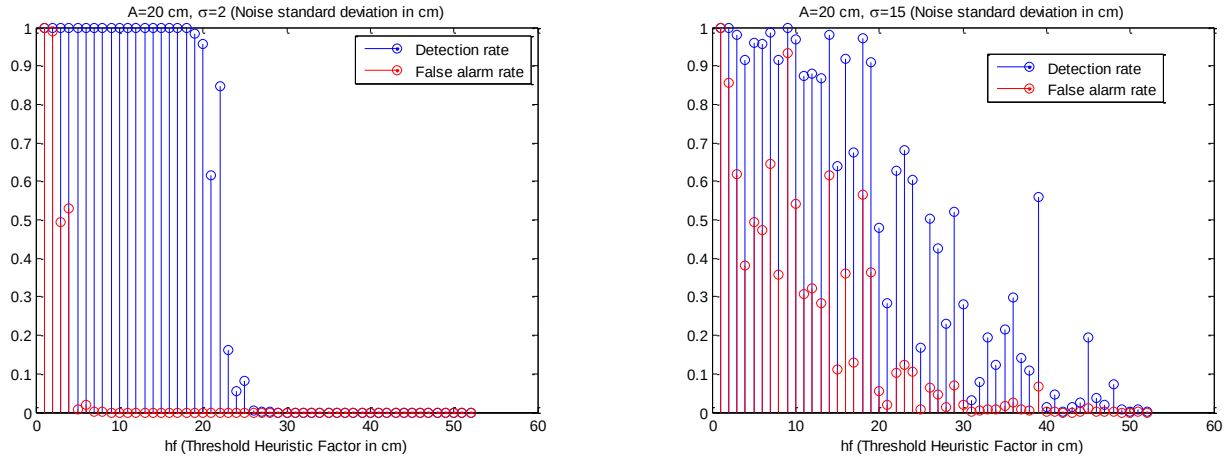


Figure 12: Influence of the noise variance given a certain event amplitude.

Next, we assess the influence of the event length in the detection performance. The results are presented in Figure 13. It can be observed that as the event length is reduced the performance gets worse. Namely, the detection rate decreases and the false alarm rate may increase.

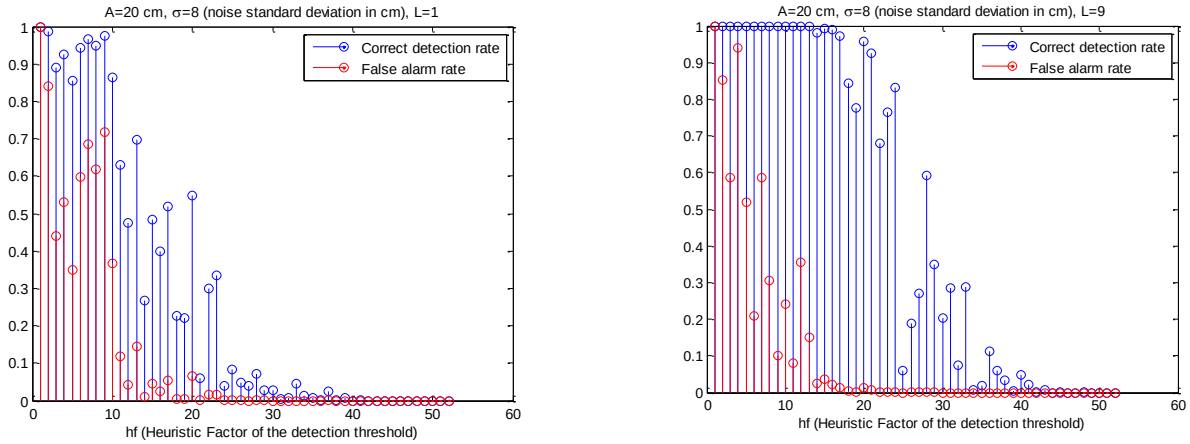
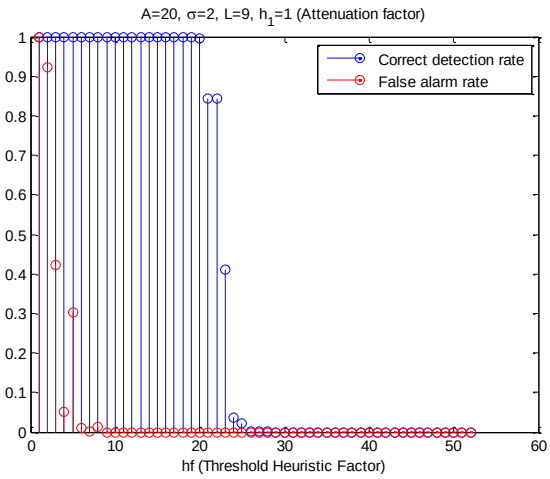


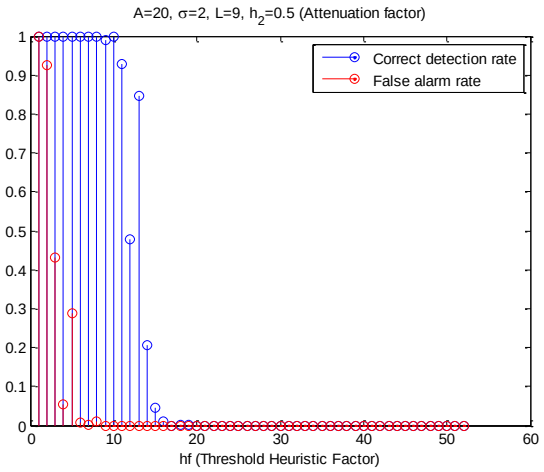
Figure 13: Influence of the event length given a certain event amplitude and noise variance.

In Figure 14 a multiple sensor scenario is considered, namely a detection system composed of three WSN nodes is considered. An event is present in the scenario with an amplitude A=20 cm and L=9 in 30% of the processing buffers (i.e. $P_{event} = 0.3$). Moreover, in the first sensor the event is observed without attenuation i.e. A=20 cm, however, in the other sensors the event is observed with an attenuation factor of $h_2 = 0.5$ and $h_2 = 0.1$. This means that the second sensor node observes an event with A=10 cm (which is half the amplitude of the actual event),

whereas the third sensor observed an event with A=2 cm. In Figure 14 the effect of the attenuation can be observed. Namely, it provokes a reduction of the margin where the detection rate is high and the false alarm is low. This implies that the selection of the heuristic threshold factor in the second sensor is more sensible. This effect worsens in the third sensor node, as only very few values of *hf* permit an acceptable detection and false alarm rates.

In Figure 15 the same simulation conditions than in Figure 14 are taken into account. The aim is to study the performance after applying the fusion of detections method described above, i.e. the k out of M rule. More specifically, k=1, which leads to a logical OR fusion rule. Recall that the fusion rule has as input data the detections of each WSN node; thereby it outputs a decision which is a logical OR among the individual decisions of each node. This rationale can be observed comparing Figure 14 and Figure 15. That is, observe that the performance of the fusion is similar to the best performance among all the individual WSN nodes, which in the case of Figure 14 corresponds to the first WSN node (i.e. the upper left plot). This shows the beneficial effects of applying a fusion rule. On the one hand, the performance of the worse WSN nodes is improved, e.g. if we had only the sensor node 3, then the detection of the event will be rather difficult as the signal is observed with a high attenuation. On the other hand, the performance of the best WSN node is almost not affected after the fusion method. Only an increase in false alarm rate is observed if the heuristic threshold factor is selected with a very small value, i.e. close to the noise floor.





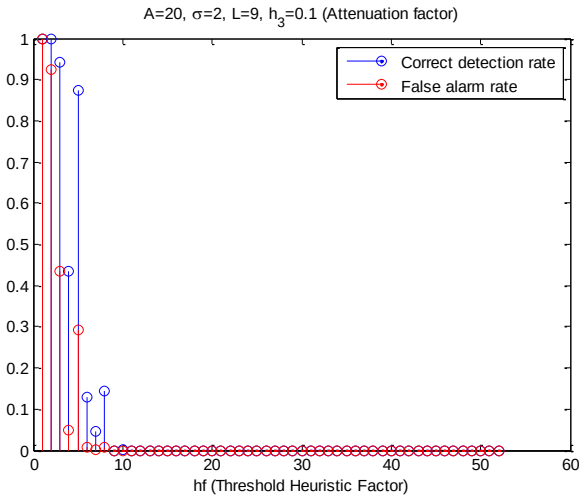


Figure 14: Influence of the attenuation factor in a multiple sensor scenario.

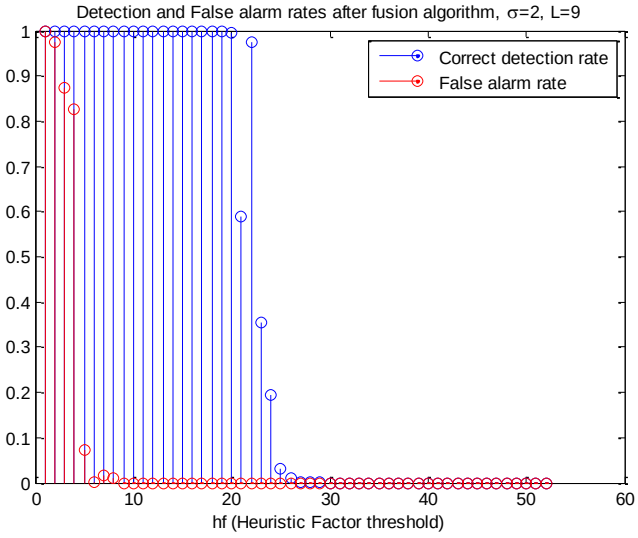


Figure 15: Detection performance after the Fusion algorithm.

2.2.2.3 Real Scenario

2.2.2.3.1 Description of the experimental setup

In order to validate the conclusions extracted from the simulated scenario CTTC has developed a platform to test real experiments. Although this deliverable is focused on Large Area Surveillance Applications, for the sake of simplicity, CTTC has developed a testing platform for a laboratory environment. The platform presented here can be dimensioned for an outdoor scenario choosing the appropriate sensors.

Figure 16 shows a block diagram of the experimental platform developed. It is constituted by 3 ultrasonic sensors, 4 Zolertia Z1 motes and a Raspberry Pi 2 Model B. In general, each one of the ultrasonic sensors detects the presence or not of someone or something in front of it. Then, each one of the motes gathers data from the sensor and sends them to the sink mote via a wireless link. Data is encapsulated in unicast packets. Then the sink mote forwards the data received to the Raspberry Pi (or fusion centre) where it is processed thanks to the fusion algorithm implemented therein.

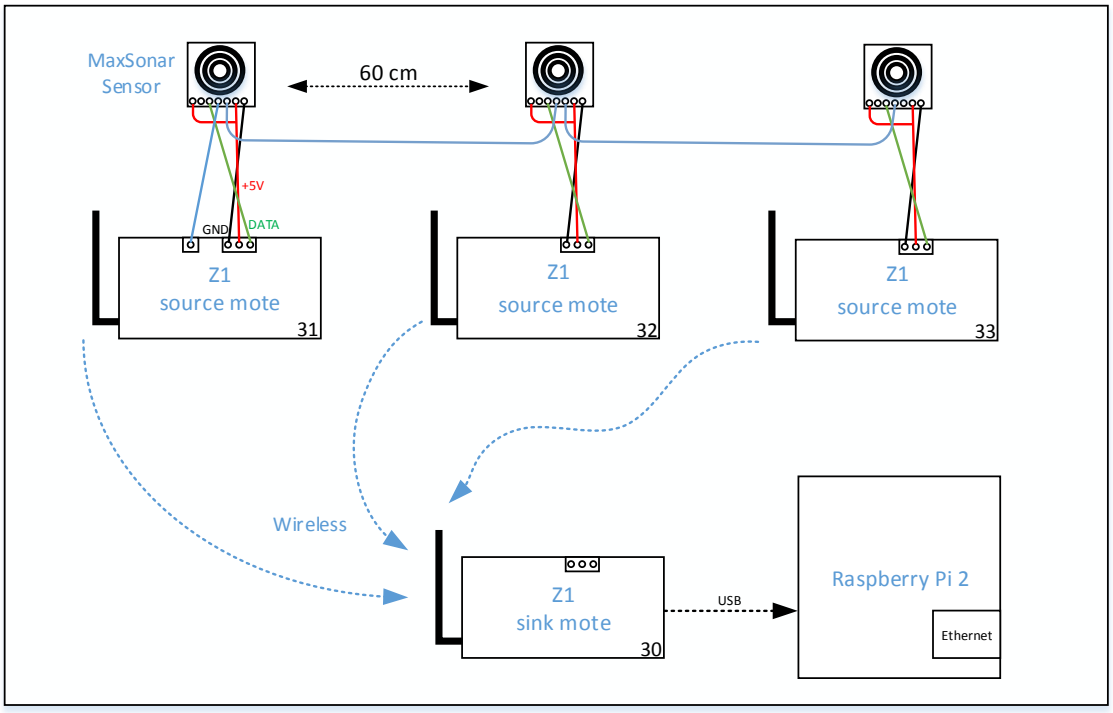
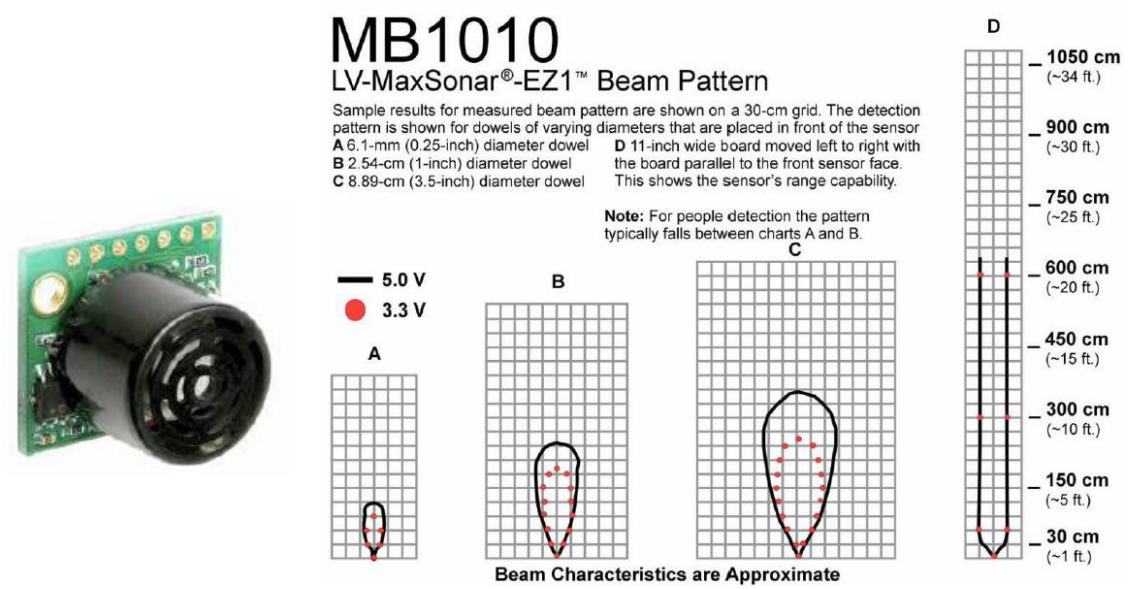


Figure 16: Experimental setup

The ultrasonic sensors chosen are the LV-MaxSonar-EZ1 model MB1010 from MaxBotix Inc. It is a low cost device with high sensitivity, a narrow beam and specially used for people and object detection. The LV-MaxSonar-EZ detects objects from 0 to 6.45 meters and provides sonar range information with 2.54 cm (1-inch) resolution. The interface output formats included are pulse width output, analog voltage output, and RS232 serial output. In our case, the analog voltage output has been chosen. Figure 17 shows a picture of the sensor and its beam pattern.



a) b)

Figure 17: LV-MaxSonar-EZ1: a) ultrasonic sensor; b) beam pattern

With the purpose of giving some redundancy to the system and to avoid situations in which someone or something is not detected by a single ultrasonic sensor, an array of sensors has been built, as it can be seen in Figure 16. When using multiple ultrasonic sensors in a single system there can be interference (cross-talk) from the other sensors. MaxBotix Inc. has engineered and supplied different solutions to solve this problem. The solution is referred to as chaining. One of the methods of chaining that work well to avoid the issue of cross-talk is the known as Analog Output Commanded Loop method (see Figure 18).

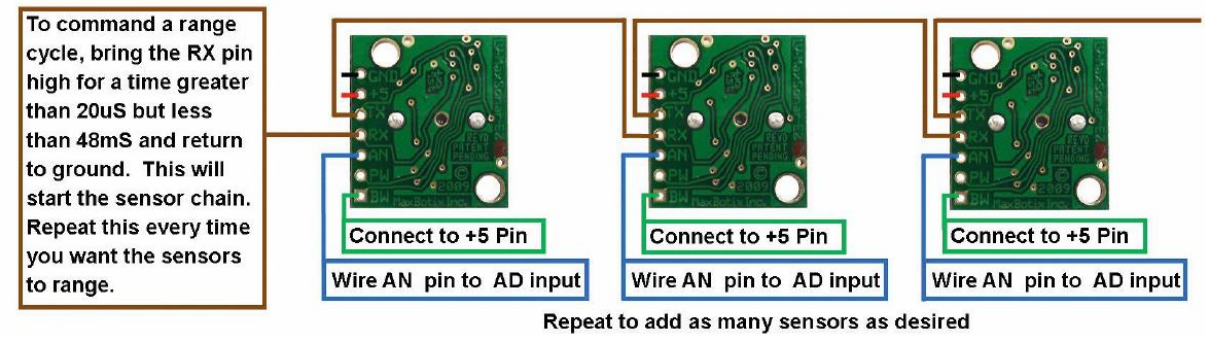
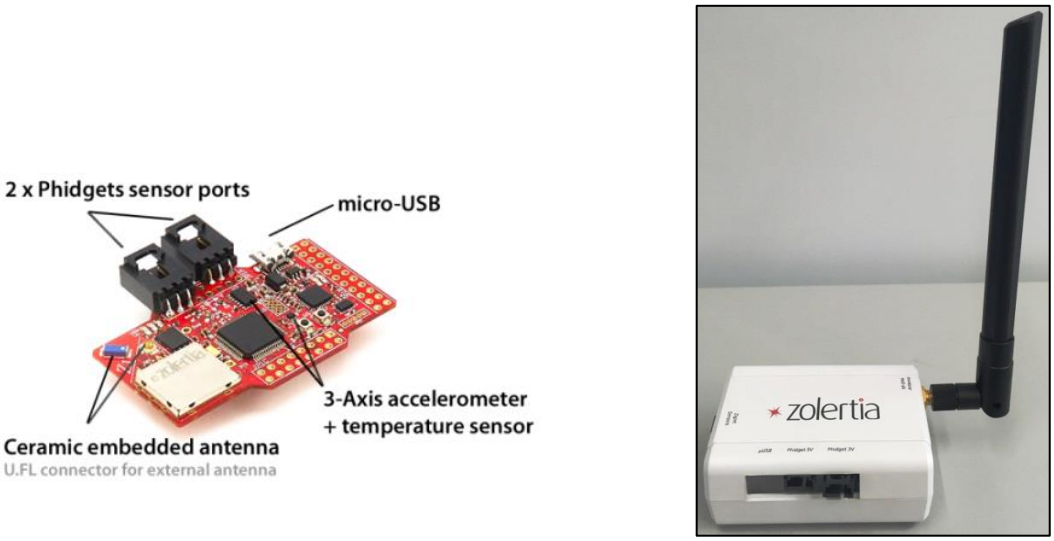


Figure 18: Analog Output Commanded Loop

The first sensor range, then trigger the next sensor to range and so on for all the sensors in the array. Once the last sensor has ranged, the array stops until the first sensor is triggered to range again. In our implementation three sensors have been used. Later it is explained how the ranging array is started. After some preliminary experiments and due to the thickness of the

beam it was decided that an appropriate distance between sensors was 60 cm. Note that in an outdoor scenario the crosstalk issue can be solved by separating the sensors with an appropriate distance as well.



a) b)

Figure 19: Zolertia Z1 mote: a) Z1 connections b) Z1 external box

Figure 19 shows in more detail the WSN nodes used in the platform. They are Z1 motes by Zolertia equipped with a second generation MSP430F2617 low-power microcontroller, which features a 16-bit RISC CPU @16MHz clock speed, a built-in clock factory calibration, an 8 KB RAM and a 92 KB Flash memory. They also include the CC2420 transceiver, which is IEEE802.15.4 compliant, operating at 2.4 GHz frequency band with a data rate of 250 Kbps. The sensors support Contiki OS, an open-source Operating System for the IoT, which connects tiny, low-cost, low-power microcontrollers to the Internet and supports IPv6 through 6Low-PAN. Each mote can operate either as a source or a sink node. The pair sensor and source mote (see Figure 20) constitutes the nodes of the platform.

The MaxSonar sensor is connected directly to the 5V Phidgets port of the source mote. It is connected through 3 coloured wires: red (power +5V), black (ground, GND) and green (analog input). Moreover, an extra wire (blue) is used to synchronize the ranging in the sensor array (see Figure 18). To start ranging, the blue cable is connected to a certain pin of the Z1 mote whose level of voltage can be modified thanks to the following instructions:

```
relay_on();      // Bring the pin of the Z1 to HIGH
clock_wait(6);   // Wait for a multiple of ~8ms (a tick). So, we wait 6*8ms=48ms
relay_toggle(); // Change the status to LOW
```

Figure 20: Node: pair ultrasonic sensor + Z1 source mote

This generates a pulse signal which is reused to trigger the other sensors in the chain.

Finally, the data arrives to the sink mote which is connected to the Raspberry Pi via USB, as it can be seen in Figure 21. The Raspberry implements the fusion algorithm and can be controlled remotely thanks to network protocols such as Secure Shell (SSH).

Figure 21: Raspberry Pi 2 + Z1 sink mote

These are the main technical features of the Raspberry Pi Model B used for the project:

- A 900MHz quad-core ARM Cortex-A7 CPU
- 1GB RAM
- 4 USB, 1 Full HDMI and 1 Ethernet ports
- 40 GPIO pins
- Combined 3.5mm audio jack and composite video

- Camera interface (CSI) and Display interface (DSI)
- Micro SD card slot
- VideoCore IV 3D graphics core

Note: The appendix at the end of the document contains the pseudocode used to program the motes and the fusion centre.

2.2.2.3.2 Experimental results

Figure 22 shows the platform developed by CTTC to test the simulated scenarios. In front of the platform there is a wall located more than 7 meters. This platform corresponds to the block diagram explained before (see Figure 16 for more information).

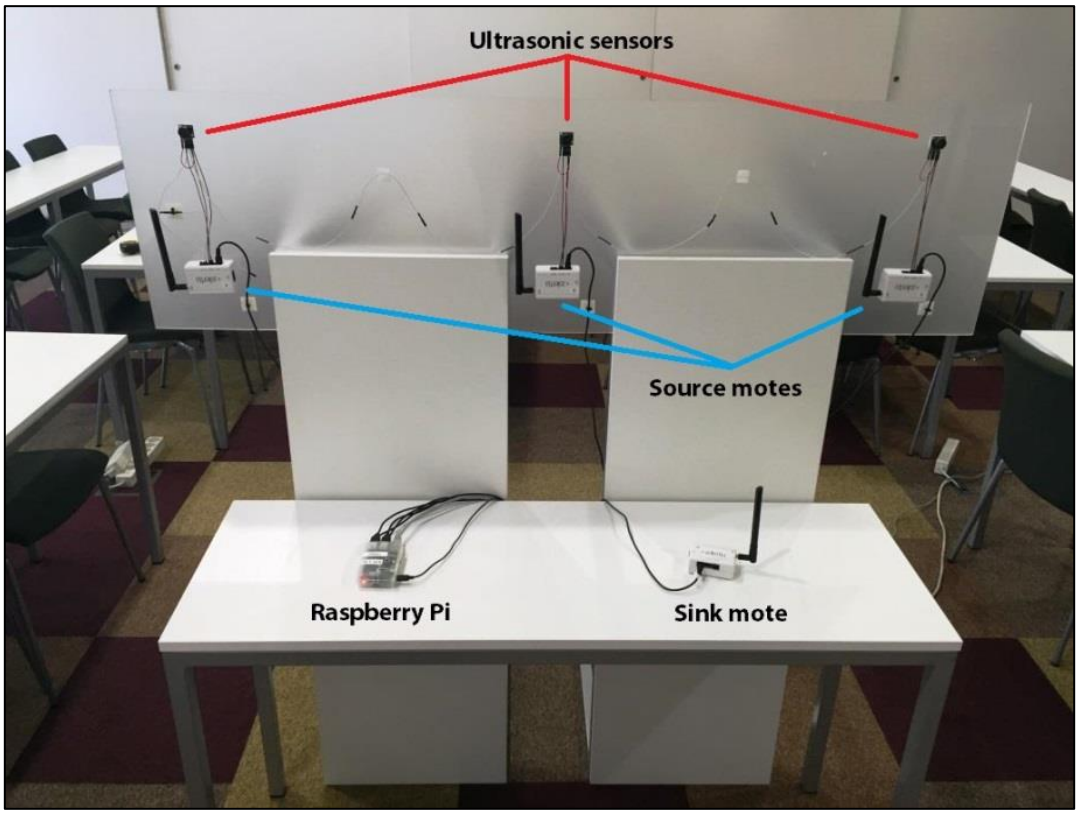


Figure 22: Platform developed

The first test consists on showing the good features of the sensors used for the experiments when we are in an only noise scenario. As it has been commented before, the LV-MaxSonar-EZ sensor detects objects from 0 to 6.45 meters. Due to this reason, Figure 23, Figure 24 and Figure 25 show the performance of the three sensors integrated into the platform. To make this measure it has been left free of objects the area (7 meters) in front of the platform. Due to a matter of graphic visualization, only the first 30cm are shown. The noise variance of the sensors is very low, as it can be appreciated in the figures.

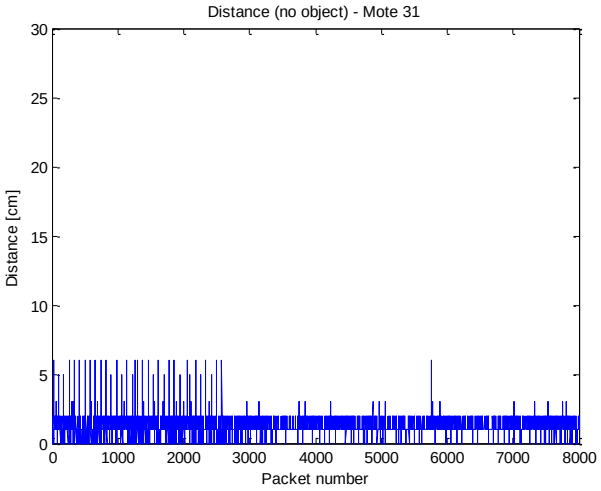


Figure 23: Output without object (Mote 31)

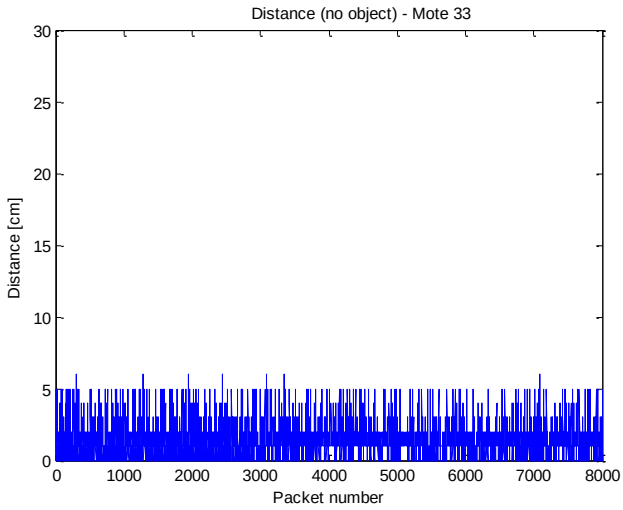


Figure 24: Output without object (Mote 33)

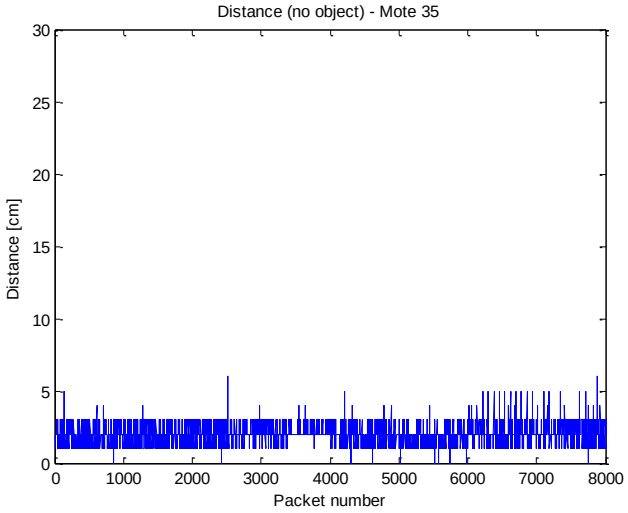


Figure 25: Output without object (Mote 35)

In order to test test the feasibility of the platform two different experiments have been implemented:

Experiment 1: Single Sensor Case. Place a narrow object in front of one of the sensors.

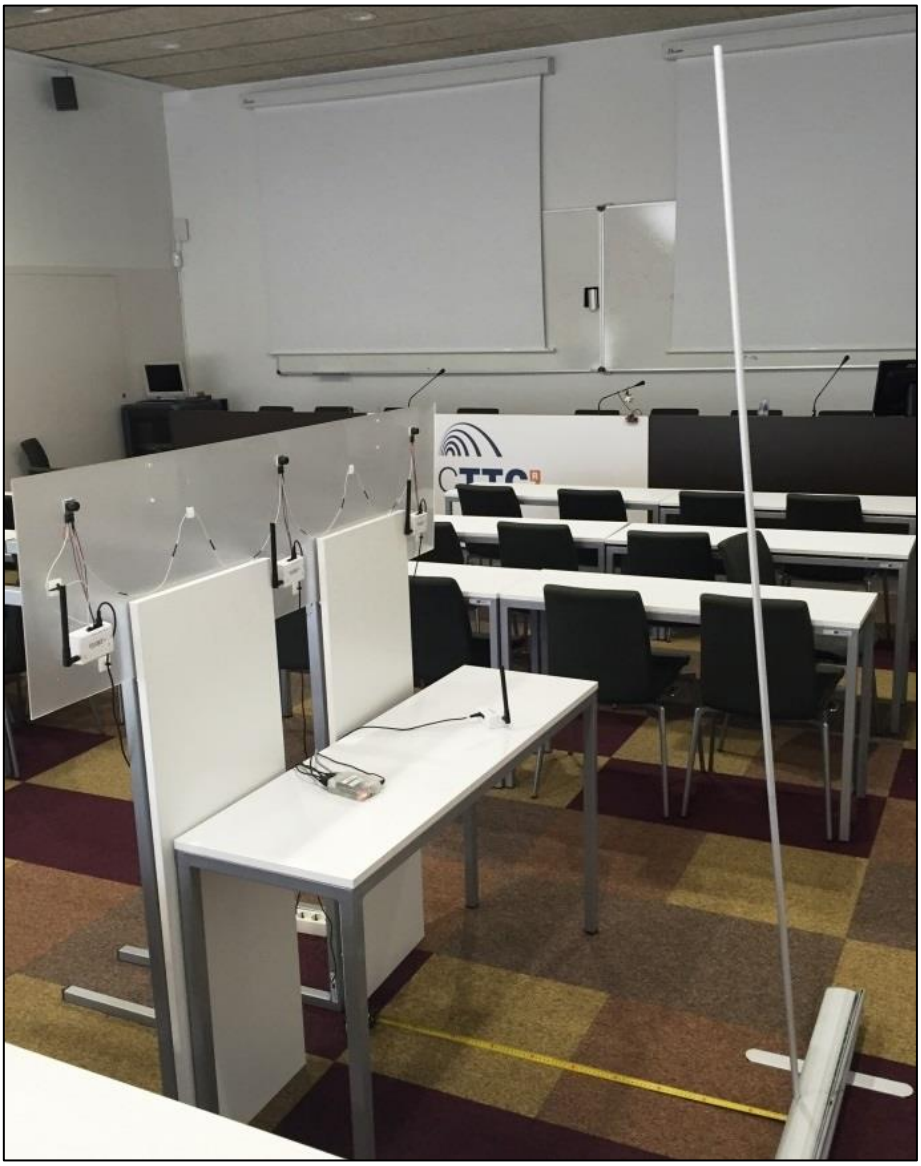


Figure 26: Experiment 1 – Narrow object

Figure 26 shows the experiment made in the Auditorium of the CTTC. Figure 27, Figure 28 and Figure 29 show the results of these experiments. Data shown in these figures is measured at the input of the sink mote which is receiving the information gathered by the source motes. Each source mote sends whether it has detected (‘1’) or not detected (‘-1’) an object/person by its corresponding ultrasonic sensor. It is important to note that a measurement is received every second from each sensor and that in the figures every packet number corresponds to a measurement. Taking that into account, between t=100 and 340 seconds a narrow object was placed 1meter in front of Mote 33 (the one in the middle of the platform). The threshold of the detectors

of each node has been set to 25 cm. As it can be seen, only the sensor of mote 33 detects the object. Therefore, the platform works perfectly well.

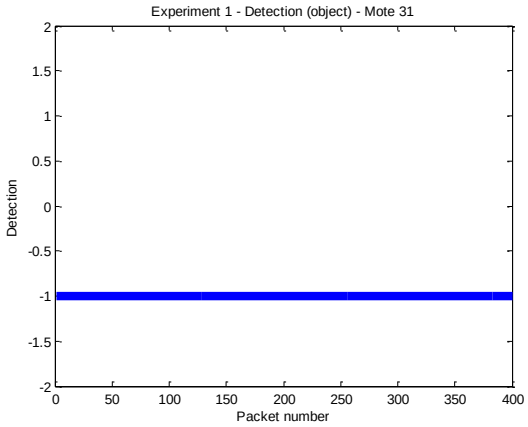


Figure 27: Experiment 1 - Mote 31

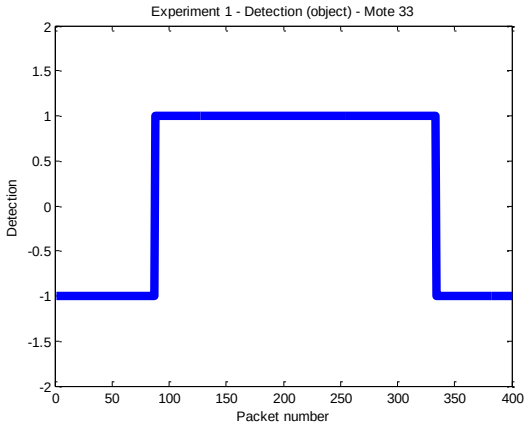


Figure 28: Experiment 1 – Mote 33

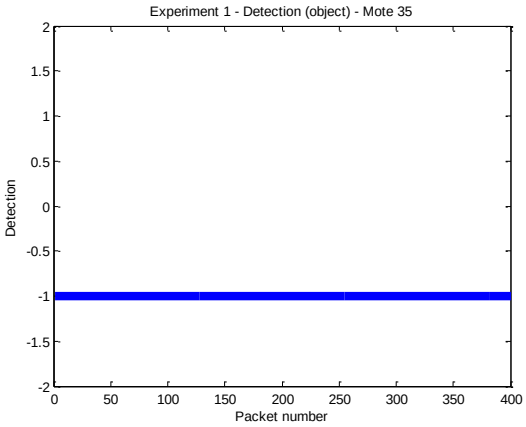


Figure 29: Experiment 1 – Mote 35

Note: The fusion centre implemented in the Raspberry Pi is not taken into account for this preliminary test.

Experiment 2: Multiple Sensor Case. Place a thick object in front of two of the sensors and on the edge of the third one.

Figure 30 illustrates the second experiment. In comparison with the previous one now the object is thicker. It is placed 1 meter in front of motes 31 and 33 but on the edge of mote 35. The object is placed at the limit of coverage, of mote 35 (sensor 3), as it can be seen in the block diagram in Figure 31. The threshold of the detectors of each node has been set to 25 cm.

Figure 30: Experiment 2 - Thick object

Figure 31 explains in a graphical manner the idea behind this experiment.

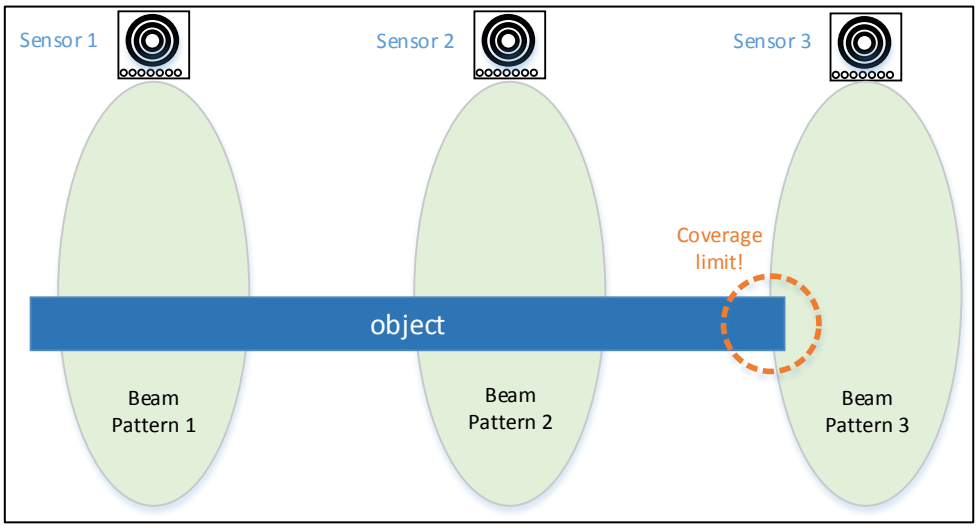


Figure 31: Experiment 2 - Coverage limit

As it was expected, motes 31 and 33 detect the object perfectly well during all the time that the experiment is performed; results are shown in Figure 32 and Figure 33, respectively. However, Figure 34 shows the detection results for mote 35, not all the time the object is detected, it can be said that the object is detected in a random manner. In the event that there had been one sensor instead of an array of three, this could cause a poor detection. This justifies the use of an array to cover a certain region. After performing the fusion of the individual detections of each sensor, 100% detection is obtained. This was the expected result since the object was placed in front of the platform. That is, the fusion algorithm implements a logical OR of the individual detections. Thereby, if one or more sensors have a detection rate of 100%, then after the fusion process the detection rate will be 100% as well.

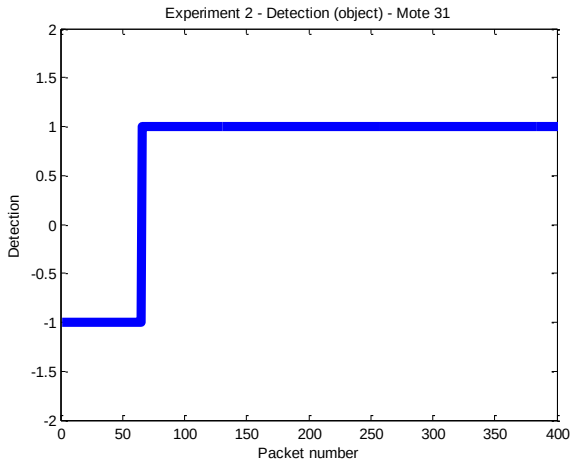


Figure 32: Experiment 2 – Mote 31

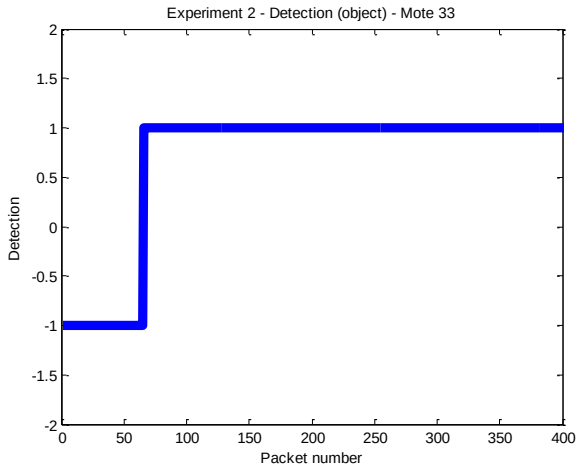


Figure 33: Experiment 2 – Mote 33

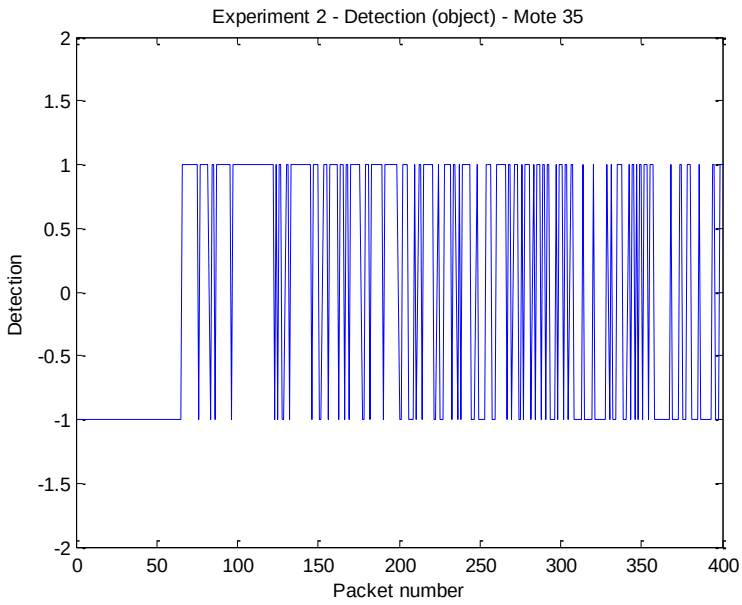


Figure 34: Experiment 2 – Mote 35

The data showed in these figures comes from the output of the motes or the input of the Raspberry Pi, the fusion centre.

Taking into account the definitions of the detection rate (*D*) factor and the false alarm (*F*) factor defined in section 2.2.2.2.1, the following calculations for this experiment can be made. The false alarm rate for the three sensors is 0% because there had been no detections when they were not expected according to the figures above. Therefore, the false alarm rate for the whole system (at the output of the fusion centre) is also 0%. This is because the noise variance is very low. Regarding the detection rate for motes 31 and 33 is 100%, all detections happened when they were expected (this is between packet number 0 and 65). At packet number 66 the object was placed in front of the platform, so from this instant until the end of the experiment (335 packets), a detection is expected for every packet:

$$D_{mote31} = 100\%$$

$$D_{mote33} = 100\%$$

$$D_{mote35} = \frac{Number\ of\ detections}{Number\ of\ detections\ expected} = \frac{220}{335} = 0.6568 \rightarrow D_{mote35}(\%) = 65.68\%$$

These results are consistent if one takes into account that the object is located right on the edge of coverage of the sensor (mote 35). However, the output of the fusion centre gives a 100%

of detection thanks to the fusion algorithm which implements a logical OR of the individual detection rates:

$$D_{mote31}\ OR\ D_{mote33}\ OR\ D_{mote35} = 100\%$$

Experiment 3: Progressively increase the threshold. False alarm study.

Figure 23, Figure 24 and Figure 25 showed the noise level of the three sensors integrated into the platform. Here we want to study what happens when the threshold is incremented by 1 (from 0 to 14cm) every 20 packets received without objects located in front of the platform, as it can be seen in Figure 35, Figure 36 and Figure 37 which show the output of each one of these sensors. Moreover, Figure 38 shows the output of the fusion centre (the result of the logical OR implemented).

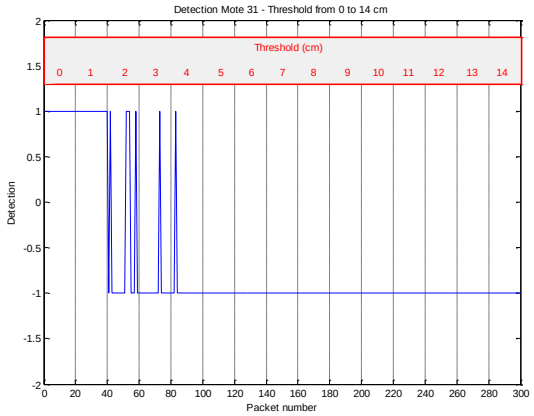


Figure 35: Experiment 3 – Mote 31

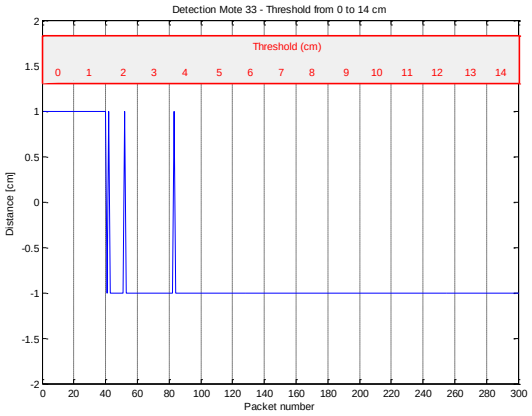


Figure 36: Experiment 3 – Mote 33

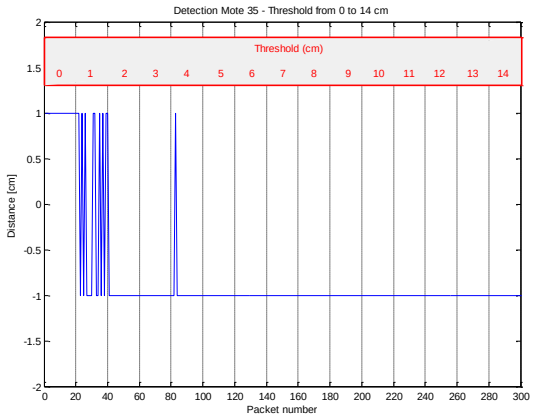


Figure 37: Experiment 3 – Mote 35

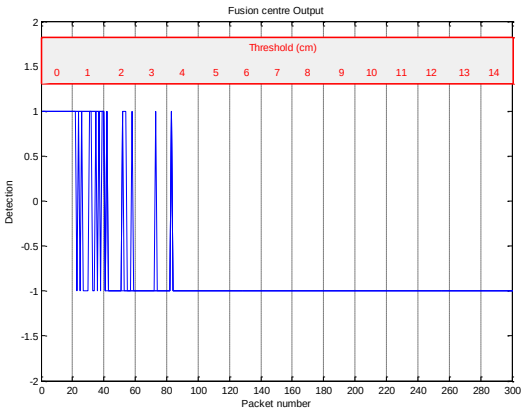


Figure 38: Experiment 3 – Fusion centre

The expected output for this experiment is '-1' which means that no detection has occurred. But, as it can be seen in the figures above, some false alarms appears when we are working with a low threshold (between 0 and 4 cm). Table 8 summarizes the results for this experiment.

Table 8: False alarm rate

Threshold (cm)	Mote 31	Mote 33	Mote 35	Fusin Centre
0	100%	100%	100%	100%
1	100%	100%	50%	100%
2	25%	10%	0%	25%
3	5%	0%	0%	5%
4	5%	5%	5%	5%
5-14	0%	0%	0%	0%

For thresholds 0 and 1cm most of the detections are false alarms, no detections were expected. As a conclusion, a threshold greater than 4 is required to have a good performance.

Finally, in order to demonstrate that the system reacts as expected, Figure 39 shows the output console of the fusion centre with the final decision taken by the detector implemented.

The figure shows the four different possibilities: no detection or detection monitored by 1, 2 or 3 sensors.

Figure 39: Console output (Raspberry Pi – Fusion centre)

2.2.3 Communication Infrastructure Simulation

2.2.4 Cognitive and Perceptive Cameras - Systems for Smart Facility Management

Facilities management is gaining increasing recognition as a significant contributor to the overall effectiveness of many organisations. Smart Facility and Building Management (SF&BM) generally involves a number of disciplines and services. The most general description to identify the market segment is understanding Smart F&BM as integrated management process that considers people, process and place in organisational context, being focused in the design and improvement of intelligent buildings (IB) and the coordination and optimization of several domains: facilities, life security, physical security and information technology. In this context, companies are becoming more interested in exploring opportunities to consolidate multiple services from single suppliers as a way of improving value. There is a significant consolidation opportunity for service providers able to deliver an integrated solution.

The motivation for this use case was to test and iteratively improve the approach (together with the use case in T5.3 Cognitive and Perceptive Vision Systems for Smart Building Scenario), in order to identify a minimum viable service (MVS) that can be provided to different clients as a comprehensive solution within the Smart Facilities and Smart Buildings Management domain.

The objective of the field test was to evaluate the potential use of a CPVS to provide different functions/profiles depending of different situations. That is, to explore the potential of COPCAMS approach, -with embedded and powerful vision systems- to sense the surrounding environment, and react to changes. In this case, the field test aims to explore the possibility of a COPCAMS system that is initially working in "asset recognition mode" to change to "surveillance mode" to detect intrusions in the specific zone. This can be used in different environments, as public or industrial facilities to control different assets (trolleys in airports, containers in maritime cargo terminals, special vehicles in public/private facilities e.g.).

Having as reference the algorithms initially defined in WP3, the iterative process has led to a set of final components, implemented in the Application Support Layer -using the component-based interfaces approach-, that have been validated in the field tests -analysing its performance with different CPU/GPU configurations-, that address two fundamental needs in Smart Facilities & Building Management:

- Asset Detection.
- Detection of Intrusions in Outdoor Work Area.

2.2.4.1 Setup description

The setup of the system has been as follows: A COPCAMS platform (PC+GPGPU) with a single camera has been placed to monitor a working area. The platform has been configured with the specific COPCAMS components, that process the input from the camera, and register the results, in order to be sent to a main station that could feed an asset management solution (as the one provided by CCTL for the management of facilities -ServiceONE®-,) a dashboard or a decision-making system.

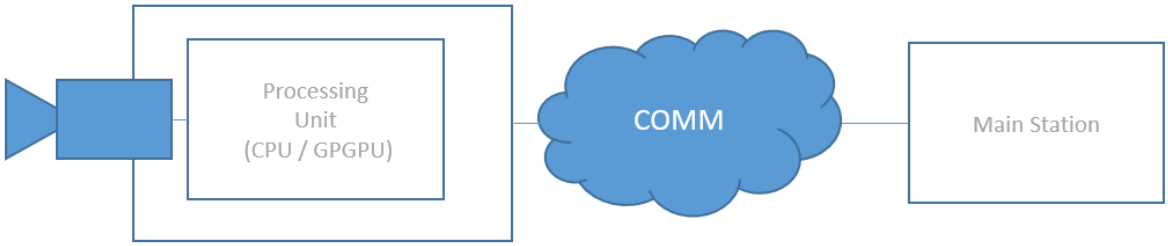


Figure 40 CCTL Setup Description

The described setup has taken place in an outdoor controlled area. We have assumed that the system is correctly installed to cover the monitoring area and the camera is calibrated.

2.2.4.2 COPCAMS modules involved

For the *CPVS for Facility Management* field test, different candidate algorithms have been developed, and analysed. An iterative process has led to a set of final components. Within the Application Support Layer of the COPCAMS' middleware, CCTL has generated specific features for human detection, and asset detection, and implemented the respective services through a component-based approach. This approach -based in a basic, common interface- facilitates the provision of different CV components that can be used in high-level applications. To generate the specific outcomes for analysing the different configurations, specific components have been developed:

- ***CPVS for Smart Facility Management:*** This is the main component. It orchestrate the communication with the Main Station, and enables the activate services for asset detection, and detection of human intrusions.
- ***Asset Detection:*** This is a C++ component, that registers data captured in real-time, to evaluate the performance of the service.
- ***Detection of Human Intrusions:*** This is a C++ component that registers data captured in real-time, to evaluate the performance of the service.

The final components of this demo make an intensive use of OpenCV 3.0, that are well optimized for GPGPU architectures, and therefore the usage of WP2 tools (Kommentator and PPCG) has not been included**.**

2.2.4.3 Scenario

The field test scenario has been performed in an outdoor, controlled area. The system is initially set up to identify specific assets. These assets have been identified thanks to a specific image pattern, and remain stopped during a timeframe of 2-4 seconds, simulating a routine control.

On a particular moment (triggered by the end of working time or by a specific simulated alarm that will be captured by the system), the system has been required to change to surveillance mode, and detect human intrusions in that specific zone. The results have been registered, in order to be sent to a main station, that could feed a facility management system as ServiceONE®.

Asset Detection

This feature is focused in exploring the potential of COPCAMS for an enhanced asset management that can be used in different environments, as public or industrial facilities to control different assets (trolleys in airports, containers in maritime cargo terminals, special vehicles in public/private facilities e.g.). The component is based in the Haar classifier- that has been trained to provide a better accuracy. The component has been then integrated in a high-level application.

Figure 41 asset identified thanks to a specific image pattern

Person Detection

This feature is shared with the "person detection" component provided in CPVS for Smart Building Scenario, in order to test the flexibility and composability of the system in different environments.

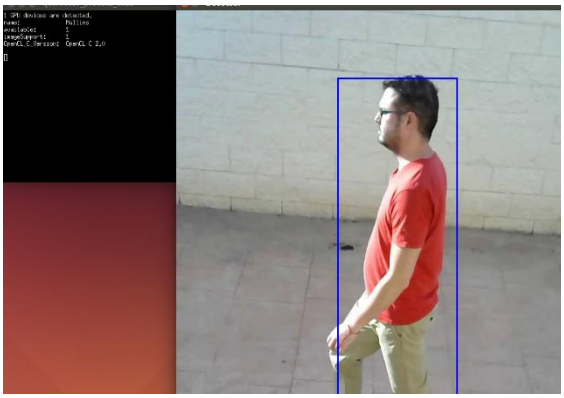


Figure 42 person detected

2.2.4.4 Outcomes of the experiments

The analysis of the outcomes has been focused in evaluating the performance of the solution in two different versions: A CPU configuration (AMD A8 7410@2.2GHz) and a GPGPU for embedded applications (AMD Radeon R5 M330), to evaluate the enhanced capabilities of a multi-core platform, that can be provided to different clients as a comprehensive solution within the Smart Facilities and Smart Buildings Management domain, taking into account performance, cost, efficiency and deployment requirements.

Asset Detection

The GPGPU configuration speed-up the asset detection. Therefore, the accuracy is highly improved. Below as follows, the results are shown:

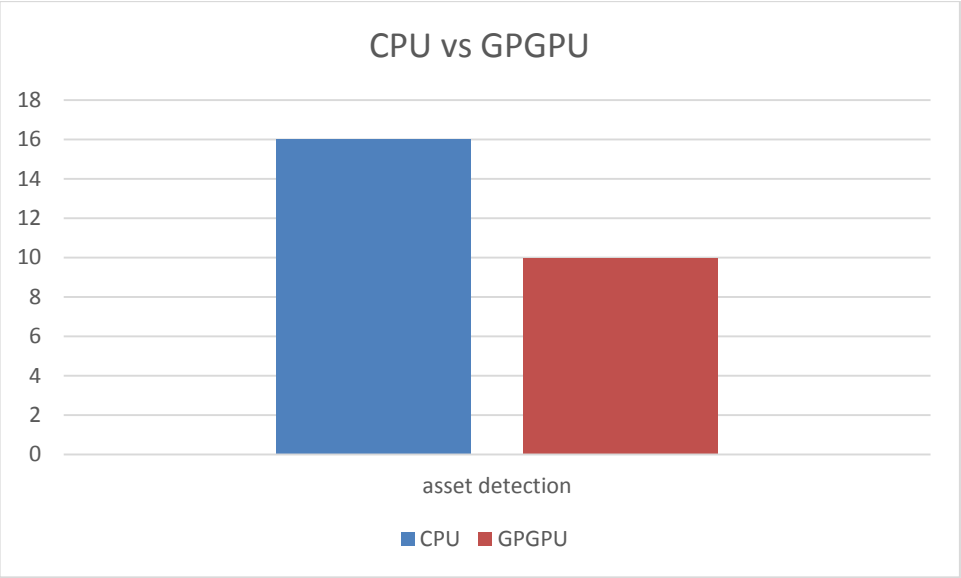


Figure 43 Asset Detection: CPU/GPGPU comparison

Person Detection

The GPGPU configuration speed-up the asset detection. Therefore, the accuracy is highly improved. Below as follows, the results are shown:

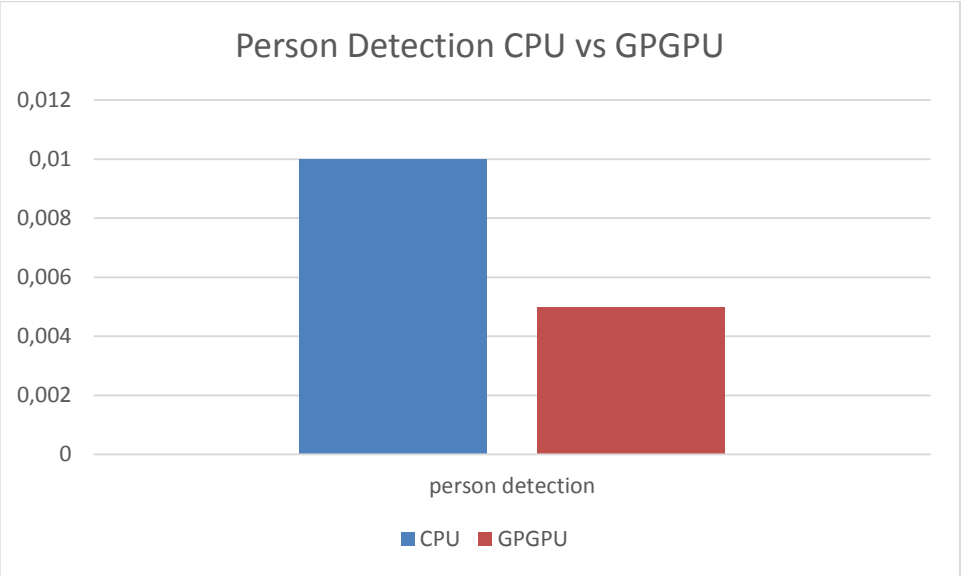


Figure 44 Person Detection CPU/GPGPU comparison

During this field test, the combination of sensing technologies with the already consolidated need of retail market for security systems has been validated. The analysis of images, usually used for physical security, has been included in a broader solution, to provide data and information for facility management.

These field tests have also validated the Component-Based approach, as a way to reduce barriers to product development and market entry. The outcomes of this field test provide a solution with clear competitive advantages: flexibility and higher performance, together with high composability than can by applied to different platforms in a field-proven technology.

The outcomes from COPCAMS are representing a significant step towards wider adoption of embedded vision systems within the Smart Facilities & Smart Building Management domain. With the outcomes from COPCAMS, CCTL is providing enhanced solutions, integrating smart video applications in future releases of its Infrastructure and Facility Management solutions.

As provider of its own tool for the management of facilities -ServiceONE®-, CCTL targets public and private organizations with a wide range of assets and areas that are difficult to manage due to their complexity (Banking, Hospitals, Hotels, Infrastructures, Logisctics, Pharma, Public Admin…).

CCTL is currently working in the definition of strategy for the commercialization phase, taking advantage of the flexibility provided by the new CPVS implemented in this project, to consolidate our position as provider of integrated services in Smart Facility Management domain for surveillance, energy management and enhanced building performance, and generate innovative solutions to strengthen our strategy for a wider deployment within the Smart City domain.

As an specific outcome of this work, it is remarkable the interest shown by a large firm, that will be in charge of the management of a large area including a seaport -that will be extended- in evaluating this flexible solution as an added-value to our Facility Management solution (ServiceONE®) combined with an innovative solution based on drones.

2.2.5 Face Detection System

The motivation of the ⟨HOE⟩^{2} lab. experiment based on the Face Tracing System from CEA is twofolds:

- To showcase the use of the development process, its prototype tool CanHOE2, and assess their benefits [33][34][36].
- To experiment with a runtime for the ⟨HOE⟩^{2} language and its code-production associated tool.

Another goal is to experiment with optimization and efficient code generation for computing kernels based on the advanced features of the ⟨HOE⟩^{2} language [35][37]. This experiment is

based on the JPEG image encoding algorithm and targets OpenCL for CPU+GPU PC-based platforms.

2.2.5.1 Face-Tracking System Modelling

The modelling of the Face Tracking System started from the use cases mentioned in "D5.1 – Large Area Surveillance Applications Specification". CEA applied a large portion of its ⟨HOE⟩[2] process toward the development of the system [33][34][36].

System Analysis

Figure 45 & Figure 46 provide a one to one modelling of the use cases documented in D5.1, albeit in French.

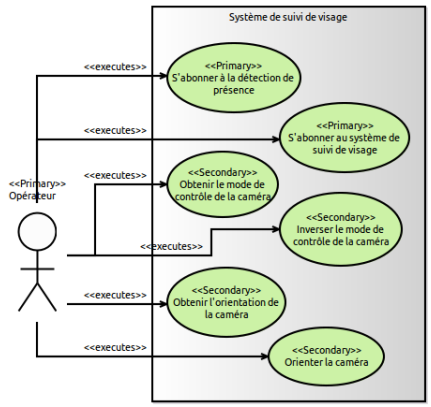


Figure 45: Use Cases of the Face Tracking System

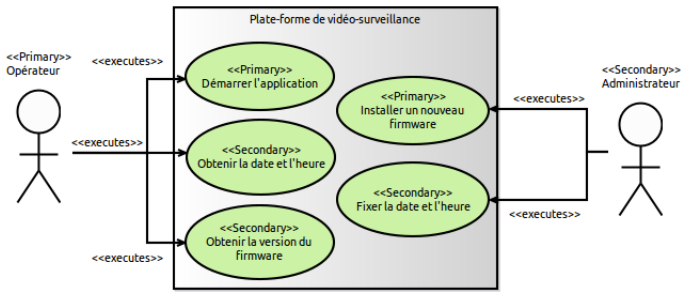


Figure 46: Use Cases of the Face Tracking Platform

For each of those use cases, a number of nominal and errors scenarios, captured as sequence diagrams have modeled the expected interactions between the actors and their system. Eventually, there are 17 scenarios spread over 11 use cases.

System Analysis

The System Analysis of the *Face Tracker Application* is shown on Figure 47. It basically captures that the business objects of the application are the *TurretController*, *FaceDetector* and *PresenceDetetcor*. These are the thee objects that allow to satisfy the scenarios of the Requirement Analysis phase, without taking into account implementation concerns.

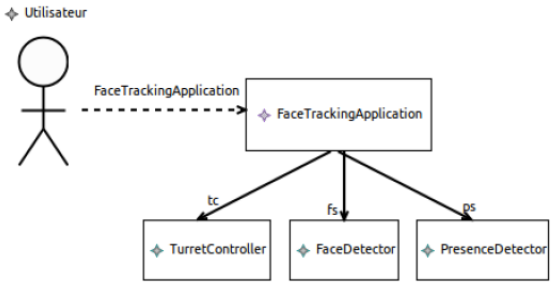


Figure 47: System Analysis of the Face Tracking Application

The System Analysis of the *Face Tracker Platform* is shown Figure 48. It presents the worlds of the platform, an abstraction of the processors, which will host application's objects. It presents also the various peripherals of each sub-platforms, *Raspberry Pi* and *Arduino*, and the way some of them are connected when the two platforms are assembled in a single one. Lastly, it presents the containers that will be used to implement the application's design objects on the platform. Those container have different coding rules on different sub-platforms and can provide access to peripherals.

For the application and the platform as a whole, we have 14 objects and 7 containers. Combined, the objects' Statecharts have 32 states and 39 transitions.

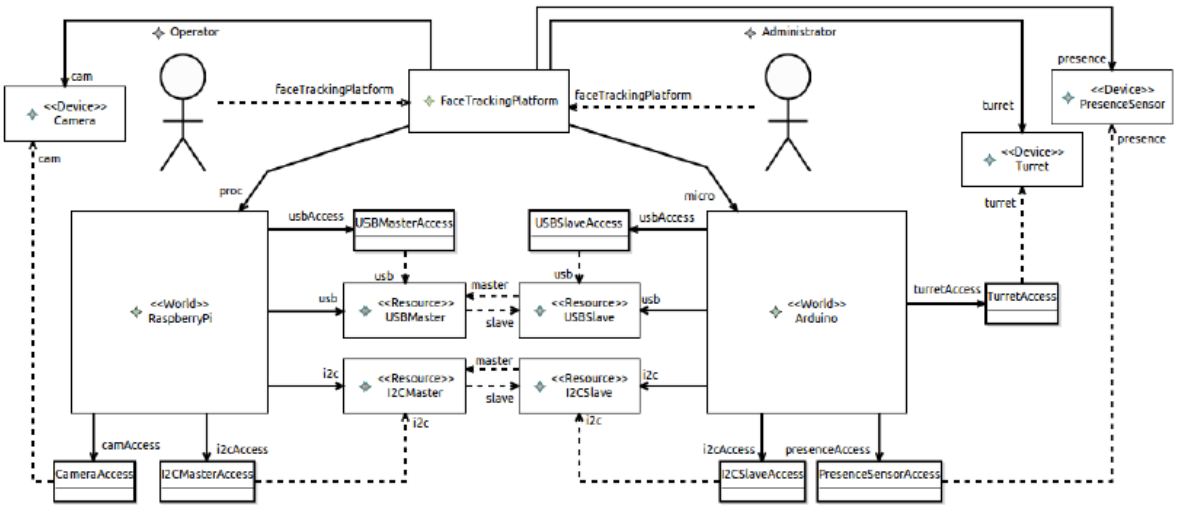


Figure 48: System Analysis of the Face Tracking Platform

System Design of the Application

Figure 49 presents the results of the System Design phase where the objects are split and distributed over the various words provided by the platform.

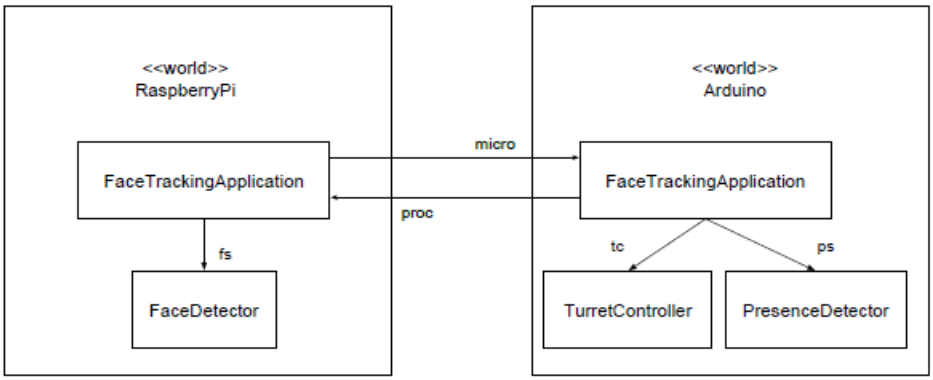


Figure 49: System Analysis of the Face Tracking Application

The associations that were already present in System Analysis are used within any of the two worlds. New associations appear to account for the fact that the *FaceTrakingApplication* has been split and distributed over the Raspberry Pi and the Arduino, and that the two pieces of what was a single object at the previous phase must now communicate together.

System Implementation of the Application

The contribution of the System Implementation is to bind split objects from the Application's System Design with a container of the world they will run on. This is captured on Figure 50. At that point the application is implemented on its platform at the model level, and the only remaining task is to apply the codding rules of the various containers and worlds to the objects to get the final application code.

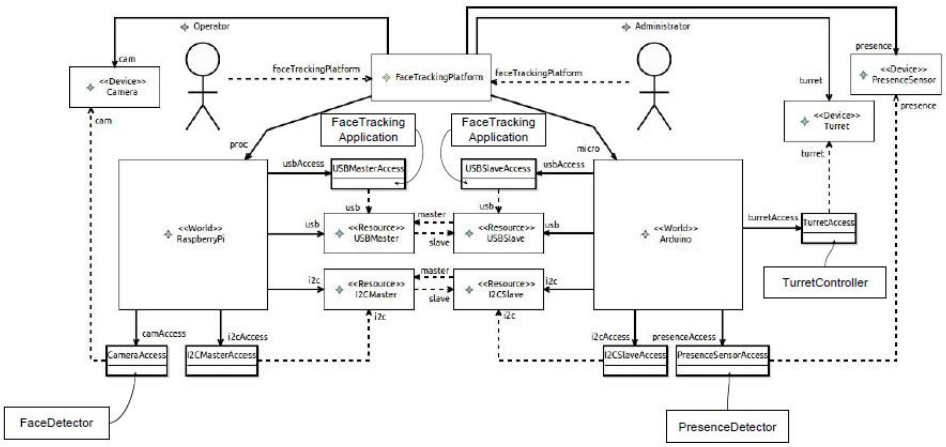


Figure 50: System Implementation of the Face Tracking Application on its Platform

At this point, the model is made of 15 objects, 7 containers. The Statecharts, combined, are made of 37 stages and 47 transitions.

Application Code generation, Platform Runtime Implementation

The code produced following the system implementation is made through a number code generators based on Acceleo[1]. The generated code of the application implement the object semantics with the help of a dedicated runtime that implements the core ⟨HOE⟩[2] semantics.

OpenCL and Parallel Code Generation from ⟨HOE⟩[2] Models

CEA has proposed an evolution of Statecharts to capture specific arithmetics and parallelism inherent to objects and associations, and developed a compilation flow that analyses the models and generate optimized OpenCL for a CPU+GPU architecture [35][37][38].

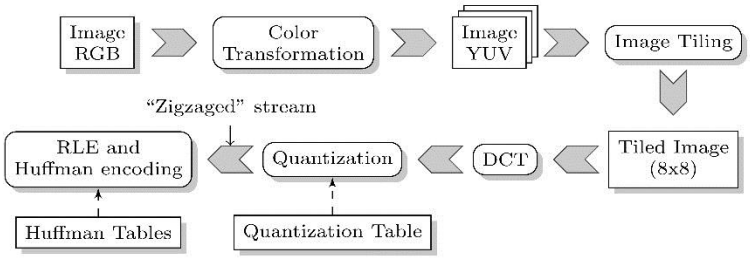


Figure 51: JPEG Algorithm

They have exercised the compilation flow on JPEG image encoding, as defined Figure 52, and produced code for PC-based CPU+GPU architecture.

Statistics & Conclusions

As of mid-September 2016, the situation is the following: The code generators are able to produce code for Ubuntu on a regular PC. The runtime is implementing the ⟨HOE⟩[2] semantics on the same Ubuntu on a regular PC. Both will be ported to Raspbian (Raspberry Pi's Debian Linux) in the near future as the real target platform is a Raspberry Pi plus an Arduino.

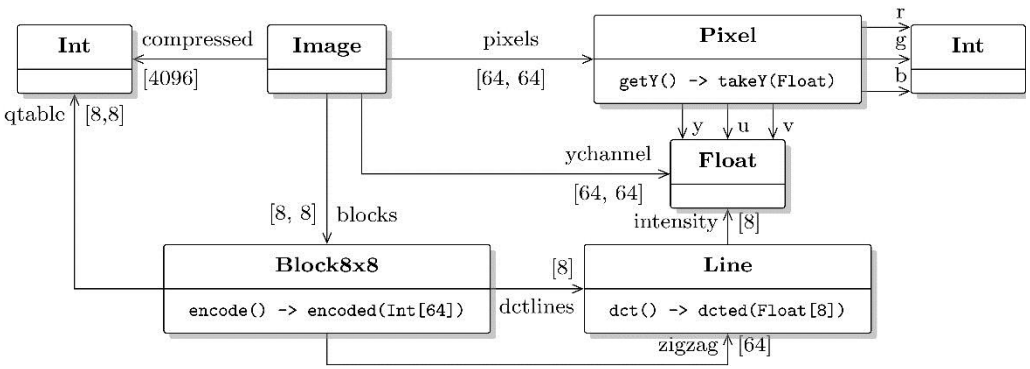


Figure 52: HOE2 Model of JPEG Image Data

The port of the code generators and the runtime will be easy as Ubuntu is a Debian derivative. The main difference will be for the containers dealing with I/Os, that will have to interface with device drivers in "`/dev`" to access the actual I/O. We have a prototype code generator targeting the Arduino that predates the current developments, and we will reuse it to complete the toolset for the whole Face Tracking Platform.

[1] A model-to-text generator framework from Eclipse, based on OMG's model-to-text language. For more details see https://eclipse.org/acceleo

The developments of the Face Tracker System and the Linux code generator were made by a junior developer, with minor contributions from more seniors personal, and will continue over the coming months. The various efforts, in *Man.day*, for the models and code to reach the currents status are:

- 50 M.d for the modelling of the Face Detection System, which is estimated to be 80% complete.
- 30 M.d for the development of the ⟨HOE⟩^{2} runtime, estimated to be 90% complete.
- 12 M.d for the Linux code generators, estimated to be 60% complete.

The application modelling will vary with the complexity of the application. The runtime will be developed only once, and will not be updated for upcoming applications. The code generators depends on the platform, and will not need to be updated for a new application.

So for a new system, the packages that will need to be updated will be the application and the code generators. If the new system is mainly a new application using the same platform, then the development effort will have to be focus on the development of the new model. If, on the other hand, the new system is the same application but working on a different platform, then the effort will focus mostly on the platform model. It may also impact the application model, but only at and below System Design.

To conclude:

We believe that the ratio of four-to-one in terms of application modelling vs. platform generator development illustrates the benefits of a clean and clear separation between application and platform modelling for embedded system developments.

With code generation from ⟨HOE⟩^{2} language, we have shown that Statechart-based modelling can be used to generate efficient parallel code for modern architecture.

2.2.6 Activity Detetection and Anonymisation system

In this part, we propose a video coding scheme with respect to privacy requirements by exploring processes of anonymisation into the video stream. The first part of this system is to detect activities from the video, and to identify the area where ‘activities’ located in the scene. This algorithm was already presented in D3.2 and D3.5, and was tested in ‘difficult’ conditions (weather, light, night) as shown in Figure 53. In the second part, the concept presented in Figure 54 and Figure 55 is a new technique for real-time reversible anonymisation, using object detection and localisation, with essential data encrypted in a compliant video bitstream.

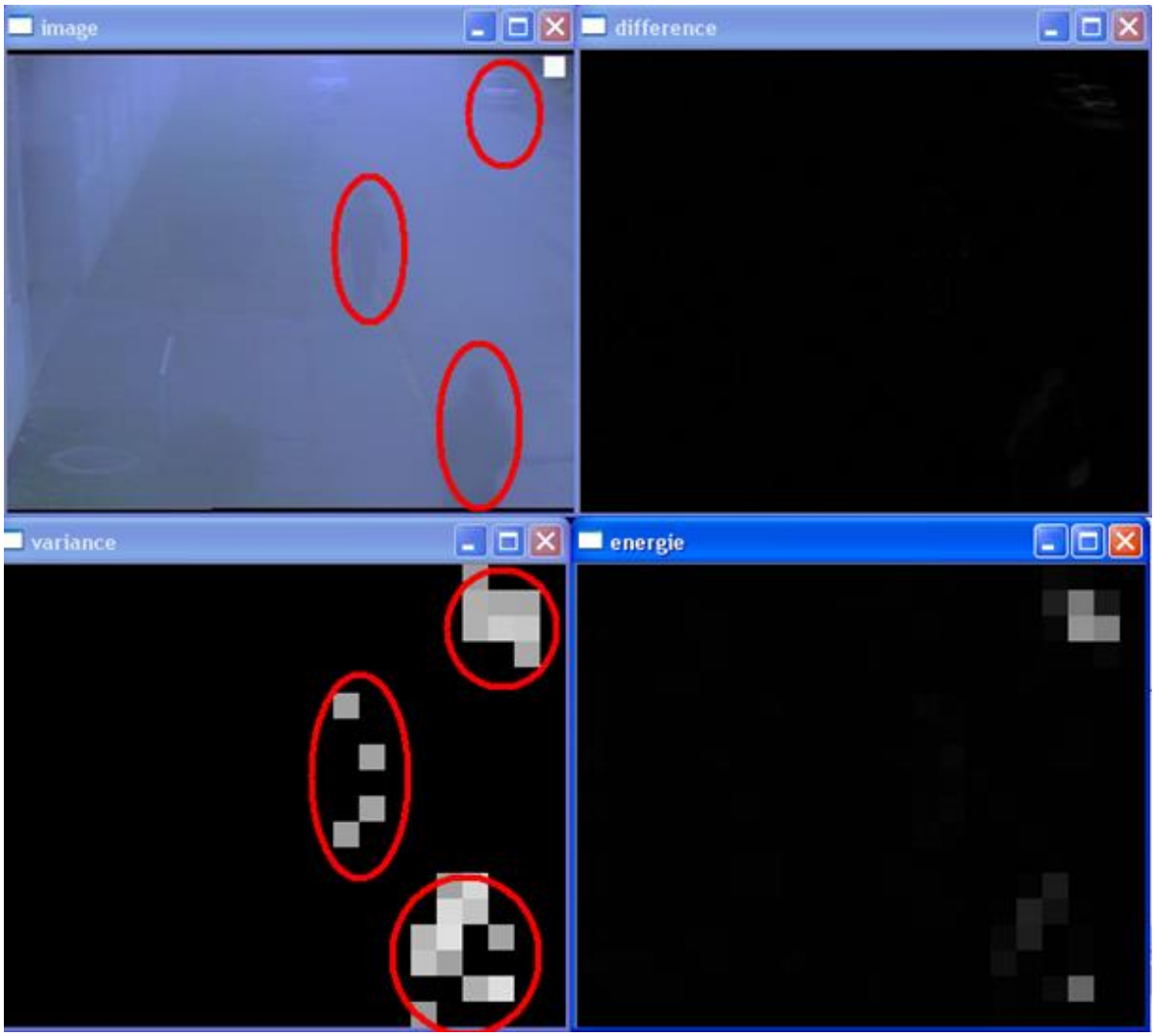


Figure 53: Activity detection algorithm

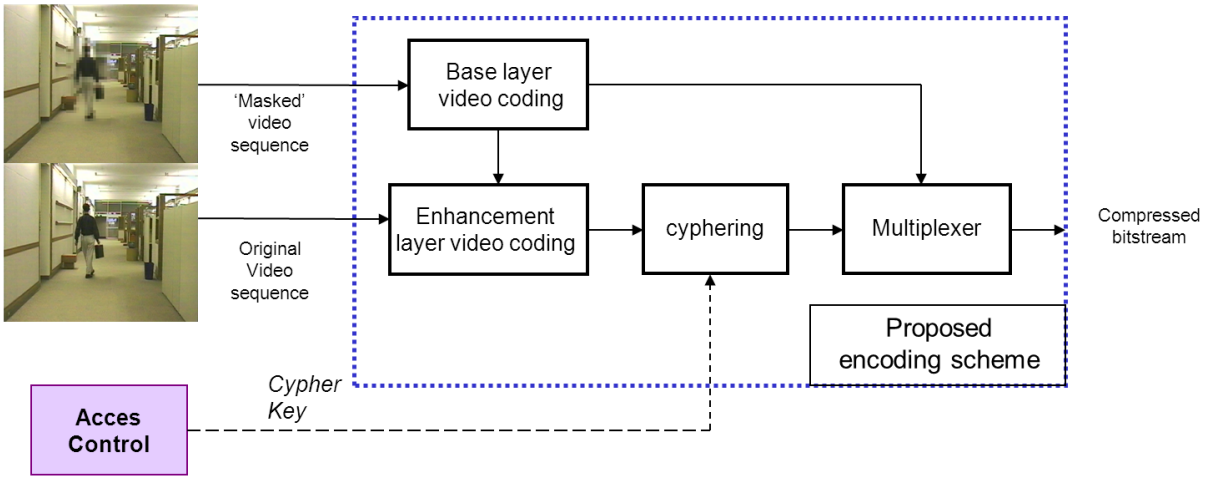


Figure 54: Proposed scheme for anonymisation at the video encoder side

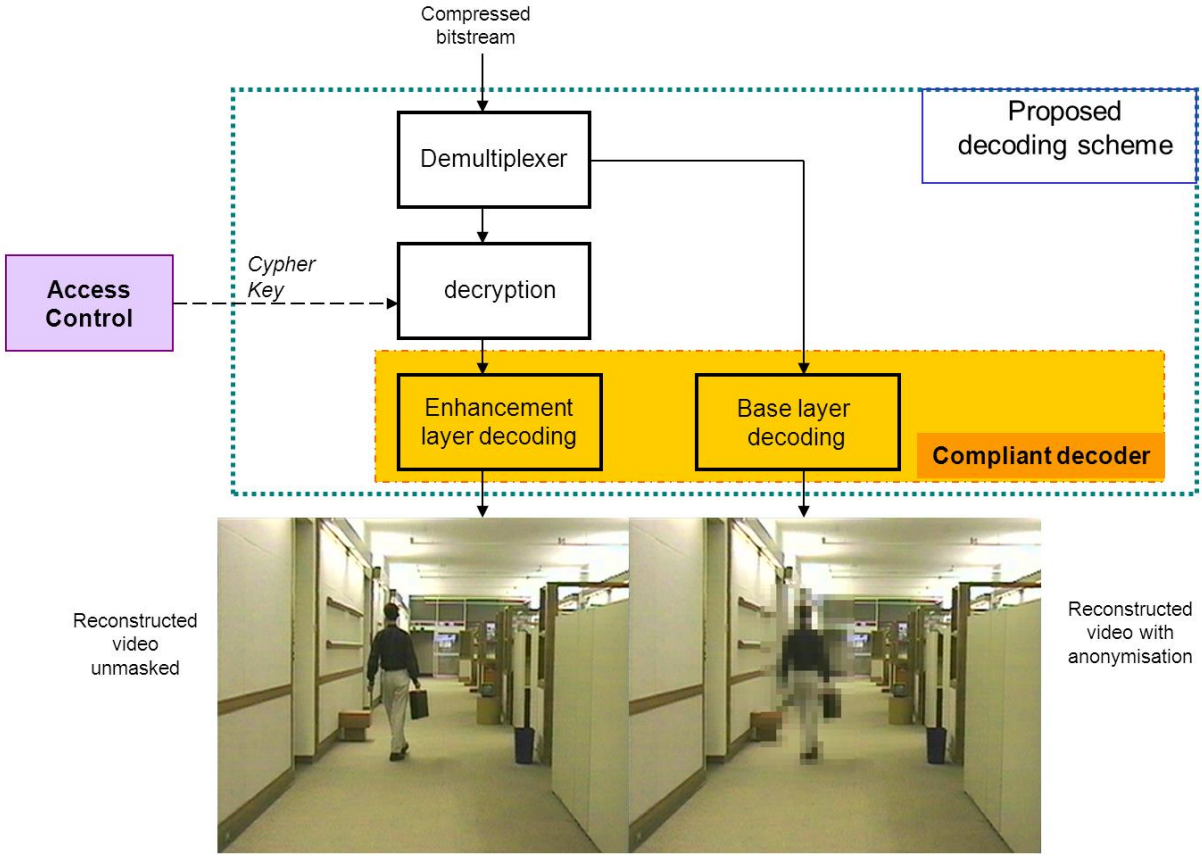


Figure 55: Proposed scheme for anonymisation at the video decoder side

The method involves compressing a masked (blurred or pixelated) video sequence using a H.264 compliant encoder in order to obtain a first compressed stream. Difference between

an original video sequence and the masked compressed sequence is compressed in order to obtain a second compressed stream by using an enhancement layer video coder as existing in H.264/MVC.

The second flow is cyphered with an access-control that is not-interpretable, such that a H.264 video decoder cannot decompress the second flow. The first and second flows are multiplexed and synchronized to obtain a single compressed bitstream.

All the algorithms were ported in IMX6 platform (see Figure 56) and optimized to work in real-time at legacy resolution for CCTV application (800x600).

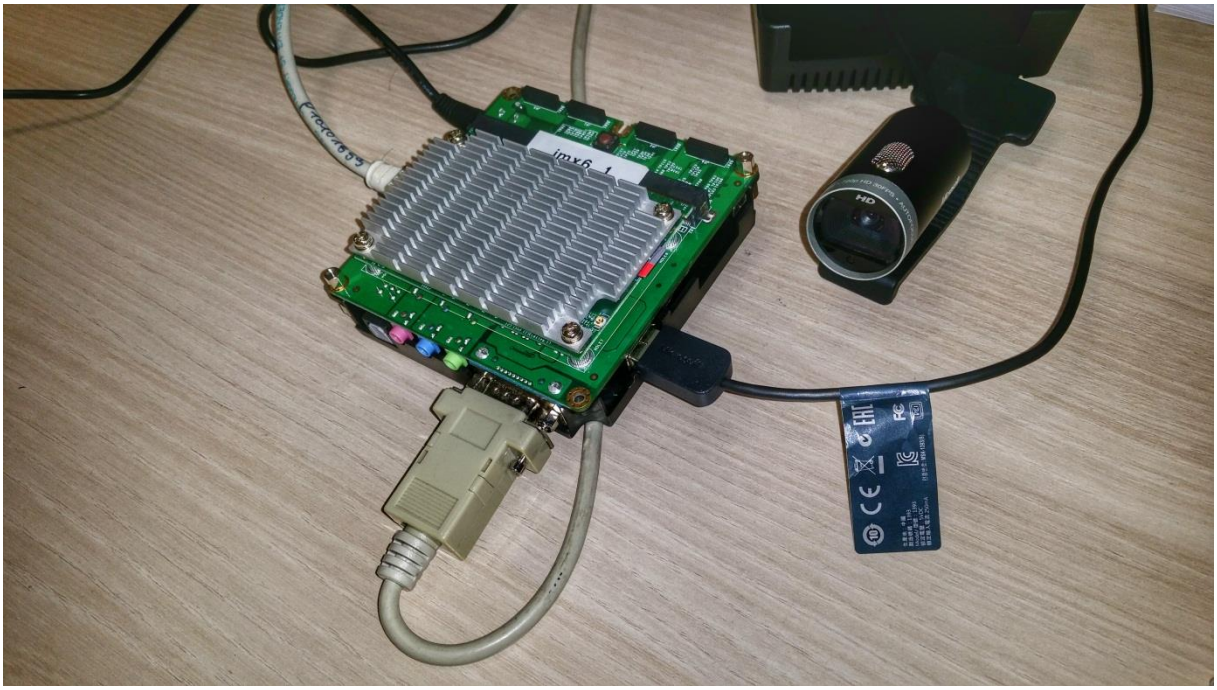


Figure 56: IMX6 platform

3 Conclusion

The demonstration activities for the large area surveillance applications are described in two categories, i.e., a field test and laboratory experiments. The methodology for evaluating field test and the metrics and measurement strategy of COPCAMS solutions on large area surveillance applications are explained. The results explored by evaluating the field test are reported.

Appendix

A. Pseudocode for detectors (at each mote)

a. Parameters

```
N=10;           %Processing buffer length (in samples).

B=10000;        %Number of processing buffers (in the case of a finite processing loop, see explanation below).

T=5;            %Number of processing buffers used for training. The training is needed to compute the floor level which is then
                used to set the detection threshold.

tb=0;%Boolean used to check whether training was carried out.

fl=0; %Mean floor level of the measured samples.

std_tmp=0;

hf=40;          %Heuristic factor for Detection Threshold (in cm below the noise level). That is the detection threshold is fl-hf. We
                assume it is equal for all the sensors, assuming that the noise level is %the same for all of them. Otherwise set a
                different threshold %for the sensors.
th=0;           %Detection Threshold

L=1;            %Number of samples that must be above threshold to decide a detection.

c=0;  %Counter for detection purposes
```

b. Detection loop

```
    while (true)   %Processing loop. Here we've two options. Infinite loop or finite loop with a given number of processing buffers. In
the latter case the code should be for(i=1:B) … end

    if(tb==0)               %Training. We needed it to compute the "floor" of the %measurements, which is then used for detection pur-
                            poses.

            while i<N*T  %Wait until we've enough samples for training.
                        fl=fl+sample;           %Iteration to compute mean floor level
                        std_tmp=std_tmp+sample^2; %Iteration to compute std level

                        i=i+1;
            end
            i=0;
          fl=fl/(N*T);
            std_tmp=std_tmp/N;
            std=std_tmp-fl^2;
            tb=1;
            th=fl-hf;
    end

    while(i<N)              %Wait until we have the N samples needed for the processing buffer.
        i=i+1;
                        if(x<th)
                                    %Counter of number of measurement
                                    %Samples below threshold
                         c=c+1;
                        end
    end
    i=0;
    if(c>=L)                %Detection
      D=1;
    else
      D=-1;
    end
    c=0;
    send(D); % Send the detection to fusion centre
    end
```

B. Pseudocode for fusion of detections (at Raspberry Pi)

a. Parameters

```
M=3;%Number of sensors

k=1;        %This is the parameter of the k out of M fusion rule. That is, fusion rule decides a positive detection if at least k
            sensors %decide a positive detection. k belongs to [1,M]
```

b. Computation of fusion performance

```
while (true) % This loop should be substituted by for(i=1:B) … end if
              %we want just to receive B decisions corresponding to each %mote

    for (each mote)
            receive(Di) %Wait for detection of M sensors
    Dt=Dt+Di;
    end

    if (Dt>=2k-M)                %Fusion rule
            %Decide detection
    else
            %Decide no detection
    end
end
```

References

[1] M. Zhao, V. Saligrama, “Anomaly Detection with Score Functions on K-nearest Neighbor Graphs”, *Neural Information Processing Systems*, 2009.

[2] E. Liotou, E. Papadomichelakis, N. Passas, and L. Merakos, "Quality of experience-centric management in LTE-A mobile networks: The Device-to-Device communication paradigm," Sixth International Workshop on Quality of Multimedia Experience (QoMEX), pp. 135-140, September 2014.

[3] P. Callet, S. Möller and A. Perkis, "Qualinet White Paper on Definitions of Quality of Experience", European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003), March 2013.

[4] K. Zheng, X. Zhang, Q. Zheng, W. Xiang, and L. Hanzo, "Quality-of-experience assessment and its application to video services in LTE networks," Wireless Communications, IEEE , vol.22, no.1, pp. 70,78, February 2015.

[5] A. Khan, S. Lingfen, and E. Ifeachor, "QoE Prediction Model and its Application in Video Quality Adaptation Over UMTS Networks," IEEE Transactions on Multimedia, vol.14, no.2, pp. 431-442, April 2012.

[6] M. Venkataraman and M. Chatterjee, "Inferring video QoE in real time,"IEEE Network, vol.25, no.1, pp. 4-13, January-February 2011.

[7] A. Asadi, Q. Wang, and V. Mancuso, "A Survey on Device-to-Device Communication in Cellular Networks," Communications Surveys & Tutorials, IEEE, vol. 16, no. 4, pp. 1801-1819, April 2014.

[8] S. Katti, H. Rahul, H. Wenjun, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the Air: Practical Wireless Network Coding", IEEE/ACM Transactions on Networking, vol. 16, no. 3, pp. 497-510, June 2008.

[9] A. Antonopoulos, C. Verikoukis, C. Skianis, and O. B. Akan, “Energy Efficient Network Coding-based MAC for Cooperative ARQ Wireless Networks”, Ad Hoc Networks, vol. 11, no. 1, pp. 190-200, January 2013.

[10] X. Wang, J. Li, and M. Guizani, "NCAC-MAC: Network Coding Aware Cooperative Medium Access Control for Wireless Networks," IEEE Wireless Communications and Networking Conference (WCNC), pp. 1636-1641, April 2012.

[11] E. Datsika, A. Antonopoulos, N. Zorba, and C. Verikoukis, "Adaptive Cooperative Network Coding Based MAC Protocol for Device-to-Device Communication", IEEE International Conference on Communications (ICC), June 2015.

[12] “IEEE Standard for Information technology-Telecommunications and information exchange between systems, local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007), pp. 1-2793, March 2012.

[13] D. Hernando, J.E.L. de Vergara, D. Madrigal, and F. Mata, "Evaluating quality of experience in IPTV services using MPEG frame loss rate, "International Conference on Smart Communications in Network Technologies (SaCoNeT), pp. 1-5, June 2013.

[14] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, K. Schindler, “MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking,” arXiv:1504.01942, Apr. 2015

[15] E. Maggio, M. Taj, A. Cavallaro, “Efficient multi-target visual tracking using Random Finite Sets,” IEEE Trans. on Circuits and Systems for Video Technology, Vol. 18, Issue 8, Aug. 2008, pp. 1016-1027

[16] B.-N. Vo, and W.-K. Ma, “The Gaussian Mixture Probability Hypothesis Density Filter,” IEEE Trans. on Signal Processing, vol. 54, no. 11, pp. 4091–4104, Nov. 2006

[17] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” Proc. of Computer Vision and Pattern Recognition, San Diego, CA, USA, Jun. 2005.

[18] (http://www.contiki-os.org/)

[19] S.M. Kay, “Fundamentals of Statistical Signal Processing, Volume II: Detection Theory”, Prentice-Hall Ed., 1998.

[20] (http://www.maxbotix.com/Ultrasonic_Sensors/Rangefinders.htm.)

[21] Z. Chair and P.K. Varshney, “Optimal data fusion in multiple sensor detection systems”, IEEE Trans. On Aerospace and Electronic systems, vol. AES22, no.1, pp. 98-101, Jan. 1986.

[22] (http://issuu.com/zolertia/docs/z1_brochure?e=1376732/2711667)

[23] Liu, J. Wang, S. Zhu, M.Gleicher and Y. Gong, “Visual-Quality Optimizing Super Resolution”, Computer Graphics Forum, Vol. 0 (1981), Num 0, pp. 1-14, 2008.

[24] (www.mistralsolutions.com/networked-ip-video-surveillance-architecture-distributed-centralized/)

[25] Nicolas Hili, Christian Fabre, Sophie Dupuy-Chessa, and Stéphane Malfoy. Efficient Embedded System Development: A Workbench for an Integrated Methodology. In Proc. Of the 6th Embedded Real-Time Software and Systems Congress (ERTS2 2012), Toulouse, France.

[26] Nicolas Hili, Christian Fabre, Sophie Dupuy-Chessa, and Dominique Rieu. A Model-Driven Approach for Embedded System Prototyping and Design. In Proc. of The IEEE International Symposium on Rapid System Prototyping (RSP 2014), New Delhi, India.

[27] Nicolas Hili, Chrisitan Fabre, Ivan Llopard, Sophie Dupuy-Chessa, and Dominique Rieu. Model-Based Platform Composition for Embedded System Design. In Proc. of IEEE 8th International Symposium on Embedded Multicore Many-core Systems-on-Chip (MCSoC-14), University of Aizu, Japan.

[28] Ivan Llopard, Albert Cohen, Christian Fabre, and Nicolas Hili. A Parallel Action Language for Embedded Applications and Its Compilation Flow. In Proceedings of the 17th International Workshop on Software and Compilers for Embedded Systems, SCOPES ’14, pages 118–127, New York, NY, USA, 2014. ACM.

[29] E. Maggio, M. Taj, A. Cavallaro, “Efficient multi-target visual tracking using Random Finite Sets,” IEEE Trans. on Circuits and Systems for Video Technology, Vol. 18, Issue 8, Aug. 2008, pp. 1016-1027.

[30] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” Proc. of Computer Vision and Pattern Recognition, San Diego, CA, USA, Jun. 2005.

[31] http://www.mathworks.com/matlabcentral/fileexchange/24020-expectation-maximization-of-gaussian-mixture-models-via-cuda, last accessed: July 2016.

[32] K. Bernardin and R. Stiefelhagen, “Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics,” Image and Video Processing, Issue 1, Jan. 2008, pp. 1-10

[33] Nicolas Hili, Chrisitan Fabre, Ivan Llopard, Sophie Dupuy-Chessa, and Dominique Rieu. “Model-Based Platform Composition for Embedded System Design”. In: Proc. of IEEE 8th International Symposium on Embedded Multicore Many-core Systems-on-Chip (MCSoC-14). University of Aizu, Japan, Sept. 23–25, 2014. doi: 10.1109/MCSoC.2014.31.

[34] Nicolas Hili, Christian Fabre, Sophie Dupuy-Chessa, and Dominique Rieu. “A Model-Driven Approach for Embedded System Prototyping and Design”. In: Proc. of The IEEE International Symposium on Rapid System Prototyping (RSP). New Delhi, India, Oct. 16–17, 2014. doi: 10.1109/RSP.2014.6966688.

[35] Ivan Llopard, Albert Cohen, Christian Fabre, and Nicolas Hili. “A Parallel Action Language for Embedded Applications and its Compilation Flow”. In: Proc. of 17th International Workshop on Software and Compilers for Embedded Systems. Schloss Rheinfels, St. Goar, Germany, June 10–11, 2014. doi: 10.1145/2609248.2609257. url: http://www.scopesconf.org/scopes-14.

[36] Salma Bergaoui, Ivan Llopard, Nicolas Hili, Christian Fabre, and Fayçal Benaziz. “Efficient Embedded System Development: An Integrated Workbench for Modeling and Project Management”. In: Proc. of the 8th Embedded Real-Time Software and Systems Congress (ERTS2 2016). Toulouse, France, Jan. 27–29, 2016. url: https://hal-cea.archives-ouvertes.fr/cea-01236474.

[37] Ivan Llopard, Christian Fabre, and Albert Cohen. “From a Formalized Parallel Action Language to its Efficient Code Generation”. In: ACM Transactions on Embedded Computing Systems (late 2016 or early 2017), issn: 1539-9087.

[38] Ivan Llopard. “Programming Embedded Manycore: Refinement and Optimizing Compilation of a Parallel Action Language for Hierarchical State Machines”. Docteur de l’Université Pierre & Marie Curie, Paris. École doctorale informatique, télécommunications et électronique (EDITE), Apr. 26, 2016. url: https://hal.archives-ouvertes.fr/tel-01350506.