

3rd International Conference on Human System Interaction HSI 2010,
Rzeszow, Poland, May 13-15, 2010

Embedded Low-Level Video Processing for Surveillance Purposes

Ginés Benet,
José E. Simó,
Gabriela Andreu-García,
Juan Rosell
Jordi Sánchez,
Instituto de Automática e Informática Industrial, Universidad Politécnica de Valencia, Spain,
{gbenet, jsimo, gandreu}@disca.upv.es, {jarosell, [jorsanpe](mailto:jorsanpe@ai2.upv.es)}@ai2.upv.es

Abstract

This paper introduces an embedded architecture and the low-level video processing algorithms developed for an intelligent node that is a part of a distributed intelligent sensory network for surveillance purposes. In this paper, details of the architecture developed for this SENSE1 node are given, together with the low-level video processing algorithms used, as well as the results obtained after their implementation. The video board has been developed using two DSP processors for video processing tasks, as well as a FPGA dedicated to image capture (VGA size) and to dispatch them to the DSP processors. The low-level software includes acquisition, segmentation, labeling, tracking and classification of detected objects into three main categories: Person, Group and Luggage. Also, additional features are extracted from each object in the frame. The unit has to communicate the classification results and the main features obtained using XML streaming to upper levels, as well as the processed frames, using a JPEG stream. All these functionalities are currently running in the built prototypes.

Acknowledgments

This work has been partially supported by SENSE project (Specific Targeted Research Project within the Thematic priority IST 2.5.3 of the 6th Framework Program of the European Commission: IST Project 033279), and by the Spanish Government by the complementary funding TIN2007-30367-E

SENSE



Embedded Low-Level Video Processing for Surveillance Purposes

Ginés Benet, José E. Simó, Gabriela Andreu-García, Juan Rosell and Jordi Sánchez,
 Instituto de Automática e Informática Industrial,
 Universidad Politécnica de Valencia, Spain,
 {gbenet, jsimo, gandreu}@disca.upv.es, {jarosell, jorsanpe}@ai2.upv.es

Abstract—This paper introduces an embedded architecture and the low-level video processing algorithms developed for an intelligent node that is a part of a distributed intelligent sensory network for surveillance purposes. In this paper, details of the architecture developed for this SENSE¹ node are given, together with the low-level video processing algorithms used, as well as the results obtained after their implementation. The video board has been developed using two DSP processors for video processing tasks, as well as a FPGA dedicated to image capture (VGA size) and to dispatch them to the DSP processors. The low-level software includes acquisition, segmentation, labeling, tracking and classification of detected objects into three main categories: *Person*, *Group* and *Luggage*. Also, additional features are extracted from each object in the frame. The unit has to communicate the classification results and the main features obtained using XML streaming to upper levels, as well as the processed frames, using a JPEG stream. All these functionalities are currently running in the built prototypes.

Index Terms—distributed embedded systems; distributed intelligence; low-level video processing; smart cameras; tracking; low-level classification; surveillance systems.

I. INTRODUCTION

Distributed smart cameras have received increasing focus in the research community over the past years [1], [2], [3]. The notion of cameras combined with embedded computation power and interconnected through wireless communication links opens up a new realm of intelligent vision-enabled applications [4], [5].

Real-time image processing and distributed reasoning made by distributed smart cameras can not only enhance existing applications but also instigate new applications [6], [7], [8], [9], [10]. Potential application areas range from home monitoring and smart environments to security and surveillance in public or corporate buildings. Critical issues influencing the success of smart camera deployments for such applications include reliable and robust operation with as little maintenance as possible.

In this line, the SENSE project undertakes the task of developing a distributed intelligent network of sensory units that aid to describe their environment in a cooperative way. This paper describes the video node in this SENSE architecture, together with the obtained results in the current developed prototypes. In Section II, the SENSE project is summarized as well as the main components of the embedded platform. In Section III the video processing algorithms developed in the SENSE node are detailed. In Section IV, the embedded architecture that performs the

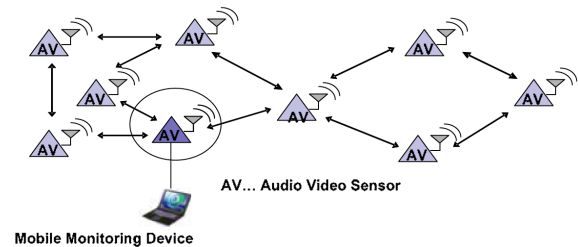


Figure 1. SENSE System Overview

low-level video processing is presented and some of their main features are discussed. In Section V, the distribution of the software between the components of the architecture aimed to achieve enough processing framerate is also presented. Finally, in the Section VI, the obtained results with real tests are presented.

II. SENSE PROJECT DESCRIPTION

A. Project Overview

SENSE [11] intends to overcome problems with current centralized networks. SENSE uses a completely decentralized approach. The system (depicted in Figure 1) consists of a number of identical, autonomous acting entities, or nodes, mounted at fixed locations.

Each entity has sensors with static sensing parameters (i.e. intelligent camera and microphones modules), gathers information from its surroundings, and interprets it. Consequently, each entity can be seen as a stand alone system. However, entities can share their knowledge with neighbouring nodes, acquiring information indirectly from other sensors. By fusing it with its own information, a global view is created autonomously.

Each node within the overall SENSE system will be able to process its own sensory data and communicate with other local nodes to build a shared understanding of objects and events, how they are related across nodes and modalities, and how they are related to the environment. Key to this distributed intelligence is the concept of a (sensor) node interacting with its neighbors.

For example, if a person walks in front of one sensor in a given direction, then that person may also walk in front of a neighbor sensor a short time later, in a direction influenced by the positioning of the node sensors relative to each other. Frequent repetition of this pattern will result in the two sensors detecting this correlation, and using it both to increase the dependability of their own observations, and to establish common views which finally should help to learn about e.g. usual paths over sensor boundaries.

¹This work has been developed under Contract N° IST Project 033279, a Specific Targeted REsearch Project within the thematic priority IST 2.5.3 of the 6th Framework Programme of the European Commission.

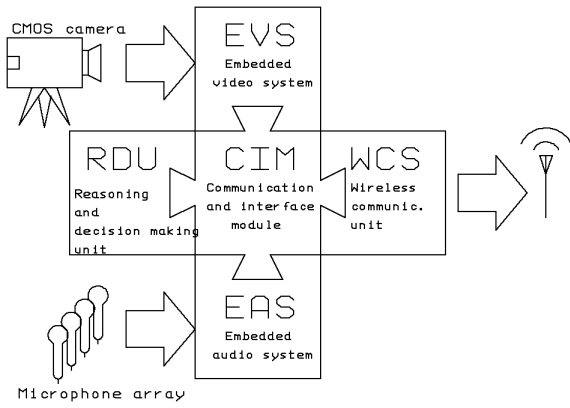


Figure 2. SENSE node design

The topology of the network is thus developed over time, and reflects the degree to which nodes can correlate their observations and thus help each other to draw conclusions about their environment, rather than a designer-defined notion of neighborhood.

B. Overall SENSE node architecture

The components of a SENSE node are organised modularly in terms of their functionality as shown in Figure 2. These components are:

RDU (Reasoning and Decision-making Unit). This module correlates information from all sensorial components in the node as well as information received from other SENSE nodes working in the neighbourhood. This module is the responsible of trigger alarms and selects which information is offered to the rest of SENSE nodes. It decides the working mode (e.g. indoors, parking area, etc.) and propagates this modality to the sensorial components.

EVS (Embedded Video Subsystem). It acquires and processes video images extracting features in a modal way. The features extracted are communicated to the RDU to enrich the perceived state of SENSE node surroundings. This component offers an interface with higher level software to control the modality, QoS and other special features.

EAS (Embedded Audio Subsystem). It acquires and processes multiple audio signals extracting features in a modal way. As the previous module, it sends its computed features to the RDU and offers an interface to higher level Software.

WCS (Wireless Communication Subsystem): This module controls the wireless communications flow in a neighbourhood of SENSE nodes.

CIM (Communications and Interface Module). This module is the hardware and software communication interface that provides the way to share information and control among modules in a SENSE node.

The hardware platform is the realization of the design in Fig.2, including some compromises due to availability of hardware components. Following the policy of using *off-the-shelf* components, FreeScale iMX31 has been selected to develop the RDU module and Blackfin DSP processors for video and audio intelligent modules.

III. LOW-LEVEL VIDEO PROCESSING ALGORITHMS DESCRIPTION

Video surveillance, either indoors or outdoors, is an active research topic in computer vision and various surveil-

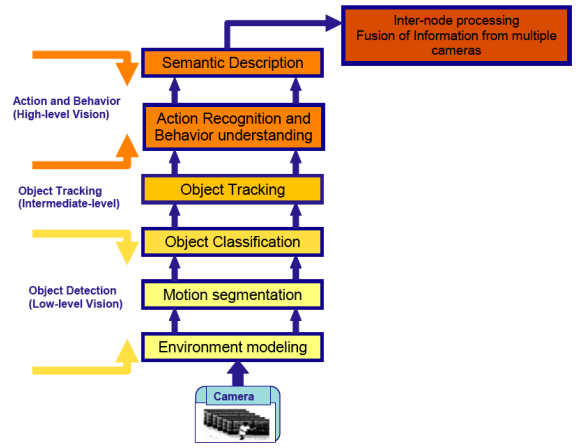


Figure 3. Video modality structure in the SENSE project. This paper focuses only on low-level vision phases, ending with Object Classification.

lance systems have been proposed in recent years: [6], [3]. The video surveillance process may be divided into the following steps: environment modeling, motion detection, object classification, tracking, behavior understanding, human identification and data fusion, (see Figure3).

From the SENSE point of view, the low level video processing sends information to upper levels in the form of Low Level Symbols (LLS). At low level, LLS are elaborated to include a draft classification into reduced pre-defined subsets matching the limited number of visual objects that can be detected.

Therefore, the low level video processing is not focused to directly solve the problem of video surveillance. Instead, the low level processing is focused on providing to high processing level with a simple object's description in the scene (including object's classification and features). In the demonstration application of SENSE, (airport surveillance), attention must be paid to individuals standing alone, groups of people and luggage. With respect to video processing, a SENSE system consists of a set of cameras running low-level vision algorithms, covering the complete airport.

The general framework proposed for video processing is shown in Figure 3. As depicted in the figure, Low Level Video Processing is responsible of object's identification and feature extraction while upper levels of semantic processing and multimodal integration are in charge of evaluating the behavior of detected objects.

Some techniques and algorithms intended for low level video processing have been evaluated in order to perform the necessary steps to detect target objects in a scene, track them and classify them. The implemented algorithms have been chosen with the aim of being fast, but without sacrificing accuracy. Also, though most of them do not require parameters, in order to allow them to be more flexible, some parameters are needed to constraint the working operation of the surveillance process. These algorithms are introduced in the next sections.

A. Environment model and adaptive background

Producing a precise background reference image is a crucial task in computer vision applications such as video

surveillance. The performance of other processes, as segmentation and tracking, included in the surveillance system depends on having a good background model. The main approaches to background modeling which can be found in literature [12], [13] are:

- Statistical approach: statistical temporal functions on a sequence of the most recent frames such as mean, mode or median.
- Parametric or non-parametric mixture model of k Gaussian distributions
- Image model approach.

This work uses the following definitions for moving objects, background pixels and so-called ghosts given in [14]:

- Moving object; a set of connected points in the input image (frame), which in comparison with the static camera, are currently characterized by non-null motion and a different video appearance from the background.
- Ghost: is a set of connected points, detected as in motion but not corresponding to any real moving object.
- Background: All pixels in every input image which neither belong to moving objects nor ghosts or their cast shadows.

Moving objects, ghosts and their cast shadows are considered as foreground objects (or regions) hereafter. Environment modeling may be also called background modeling, and both terms will appear in this paper indistinguishably. By B_i and F_i will be represented a background image and an acquired frame respectively, the subscript i indicates the instant of time when they have been computed or acquired and p_{ij} makes reference to a value pixel with image coordinates i and j .

Background modeling starts by creating a first background model B_f , which is adapted over time to cope with light changes. B_f may be obtained from a sequence of initial and consecutive input frames ($F_1, F_2, F_3 \dots F_t$) containing no moving objects, and compute the arithmetic mean, median or mode of these images

$$B_f(p_{ij}) = \text{median}(F_k(p_{ij})); k = 1..t \quad (1)$$

Evidently, simple temporal arithmetic mean does not yield correct results if the used images contain moving objects. However, a selected set of frames can produce better results and it is a simple but efficient technique. Using frames containing no moving objects must be preferred when trying to build B_f . In order to optimize resources and to obtain a representative model, a periodic but not necessarily consecutive sequence of frames as $F_1, F_{1+x}, F_{1+2x}, \dots, F_{1+tx}$ can be used instead. This approach has been graphically represented in the Figure 4.

Once the first model B_f is created, it is updated automatically after a certain amount of frames in order to introduce the illumination changes in the scene into the model. This update is performed according to:

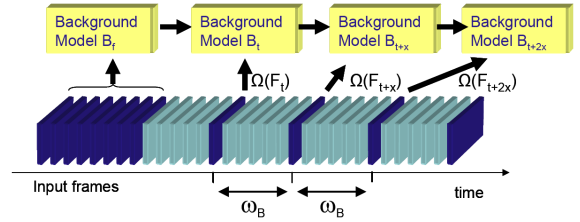


Figure 4. Background update model

$$B_{t+1}(p_{ij}) = \begin{cases} B_t(p_{ij}) + \alpha_B (F_t(p_{ij}) - B_t(p_{ij})) & \forall p_{ij} \in \Omega(F_t) \\ B_t(p_{ij}) & \forall p_{ij} \notin \Omega(F_t) \end{cases} \quad (2)$$

where B_t and B_{t+1} are the previous and newly calculated background models; $\Omega(F_t)$ is an image which only contains the background pixels of the last frame acquired and α_B is the updating rate of the background model. It is important to point out that only the pixels labeled as background pixels are updated in the background model and the other pixels are not updated. Values of α_B can vary in the range $[0..1]$, and typically a value between 0.02 and 0.05 provides good results.

Another aspect that is necessary to study is the time ω_B between two consecutive background model updates (see Figure 4). This time can be considered constant for the surveillance system or computed as a function of the frame parameters. Usually in indoors situations this time may be long, as long as illumination changes won't occur in small periods of time. The laboratory experiments show that $\omega_B = 3$ minutes would be enough in many cases. However, it is still an open line to determine whether these parameters, α_B and ω_B , should be predefined in the application or should adapt themselves to illumination evolution. The first solution would require having both as parameters of the application and has the advantage of being simple; but the second one has the advantage of flexibility, giving each node the possibility to adapt to each situation without reprogramming. In the light of SENSE, where self-configuration is an important feature, this later is the preferable alternative.

B. Local Object Tracking.

After motion detection, the next step is to track moving objects. In our case, this tracking must be considered as low-level tracking, aimed to improve the low level classification results. This must be not confused with the Object Tracking performed at the intermediate level showed in the Figure 3. First, a preprocessing of the segmentation result is performed [1], [5], [15], [8], this is done because objects may be broken into different pieces in the presence of segmentation failures. In an effort to reduce tracking errors due to this issue, an intermediate step analyzes the segmentation result and reconstructs objects according to their current position and their spatial relationships with objects in the past frames. The tracking algorithms usually have considerable intersection with motion detection during processing. Tracking over time typically involves matching objects in consecutive frames using features such as points, lines or blobs. Tracking methods are divided

into four major categories: region-based tracking, active-contourbased tracking, feature-based tracking and model-based tracking [10], [16].

We have implemented a tracking system based on the bounding box region. The bounding box (*BBox*) is the minimal rectangle that contains a blob (see Figure 11). For each blob R_t^x obtained in the motion segmentation process, its *BBox* is calculated. We assume that the movement of the object in two successive frames is such that they overlap each other. Tracking is performed by checking which bounding boxes in the current frame overlap with those in the previous one. This way, we can detect easily and with a good confidence degree which blob has moved and where, without any further test. This criterion has been found to be effective in other approaches and does not require the prediction of the blob's position [1], [8]. The following situations are considered:

- i) New blob in the scene: When a blob R_t^x in the current scene does not overlap with any other R_{t-1}^y in the previous scene, it is supposed to be new blob.

$$\{let R_t^x, let \forall R_{t-1}^y if (R_t^x \cap R_{t-1}^y) = \phi\} \Rightarrow R_t^x \text{ is a new blob} \quad (3)$$

- ii) A blob leaves the scene: When a blob of the previous scene has not overlap with any other in the current scene, it is supposed to be out of the scene.

$$\{let R_{t-1}^y, let \forall R_t^x if (R_t^x \cap R_{t-1}^y) = \phi\} \Rightarrow R_{t-1}^y \text{ is a blob that leaved the scene} \quad (4)$$

- iii) Two or more blobs join: When two or more different blobs in previous frame overlapped on a single blob in the next frame, we say that they have joined. Joins happen when people are picking a suitcase up or people are crossing or joining.

$$\{let R_{t-1}^y \text{ and } R_{t-1}^s; let \forall R_t^x if (R_t^x \cap R_{t-1}^y) \neq \phi, \text{ and } (R_t^x \cap R_{t-1}^s) \neq \phi\} \Rightarrow R_{t-1}^y, R_{t-1}^s \text{ have joined in } R_t^x \quad (5)$$

- iv) Two or more blobs split: When a single blob in previous frame overlapped with two or more different blobs in current, we say that they have split. Splits happen when someone leaves a suitcase or a group of people unravels.

$$\{let R_t^x \text{ and } R_t^p; let \forall R_{t-1}^y if (R_t^x \cap R_{t-1}^y) \neq \phi, \text{ and } (R_t^p \cap R_{t-1}^y) \neq \phi\} \Rightarrow R_t^x, R_t^p \text{ split respect } R_{t-1}^y \quad (6)$$

The joint/split detection is performed with the aim of detecting blobs joining (which could correspond to a person picking up an object or several people joining into a group) or splitting (which could correspond to someone leaving a suitcase or a group of people separating). These cases are represented by several bounding boxes which overlap one with each other.

Blobs are also labeled to point out which area of the scene they are standing in. The Kalman filter is applied to each object in order to try guessing trajectories when objects cross and it becomes impossible tracking them separately. The result of the Kalman filter is used in the case that two objects join into one and separate again. The

system assumes the Kalman filter predicts accurately the position of both blobs and then distinguishes both of them according to these predictions.

Once blobs are assigned with a label (either new or coming from the previous scene) they are considered as tracked objects, (or objects, for brevity). The result of the local tracking step is a number of objects which inherit some properties from those of the previous scene related with them, or objects which are new in the scene. For each object, either new or not, a feature set and a membership probability for each class (see Figure 11) are calculated in order to ease the final classification done by higher processing levels.

C. Assigning Membership Probabilities to found Objects: Classification

Calculation of membership probability is performed for each object in each frame. Objects which had matches with objects in the previous scene, update their probabilities again to give more confidence to previous calculations. This membership assignment constitutes a low-level classification, which is passed out to the upper reasoning levels in the overall architecture, together with the rest of calculated features, in order to aid in the final classification of each object.

The main features that have been used in low-level classification are:

- *Dispersedness*: calculated as the square of the perimeter divided by the area of the blob (see [17] for more details). This feature is useful to distinguish compact blobs (such as bags and similar objects), with reduced perimeters from others with more human-like dispersed contour, having more long perimeters (such as in persons and groups).

- *Extent*: Calculated as the amount of foreground pixels of the object divided by the bounding box area. The *Extent* of a *Blob* (also known as *Filling Factor*) is somewhat similar to the *Dispersedness*, and gives higher values for class *Luggage* objects, while for *Groups* offer more reduced values and *Persons* show intermediate values.

- *Number of heads*: Number of heads of the blob. It corresponds to the number of local X-maxima in the silhouette of the blob based in the x-projection.

The *Number of heads* is a feature that intends to separate the class *Group* from the *Person* and *Luggage* classes. Intuitively, it seems evident that a person has only one head, whereas a group must have as many heads as persons contained in the group. Also, in the case of a luggage's blob, is not unusual to found one local maximum. However, this is a simplified theoretical approach, because real blobs show many defects due to noise or failures in segmentation process. These imperfections can lead us to obtain wrong head numbers. Thus, if this parameter is to be used in a classifier, a previous pre-processing of the blob is mandatory to obtain robust estimations for its value. Currently, in our approach, the upper part of the blob's silhouette is low-pass filtered to avoid noise peaks and little objects to be confused with heads. Also, these local x-maxima must have a minimum of height from the adjacent local minima (the shoulders) to be considered. Finally, these maxima must have enough blob area under

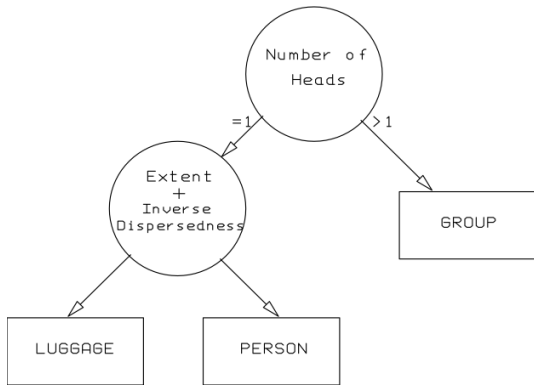


Figure 5. Decision tree used in low-level classification. First, the *Number of Heads* is used to discriminate the *Group*. After, the *Extent* and *ID* are used in a K-NN classifier to discriminate between *Luggage* and *Person*.

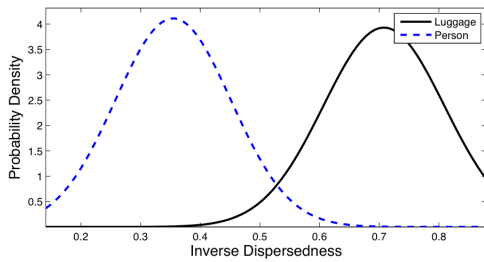


Figure 6. Statistical distributions of *ID* in *Luggage* and *Person*.

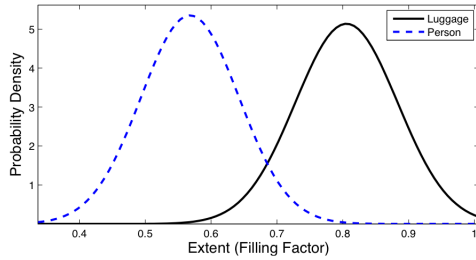


Figure 7. Statistical distributions of *Extent* feature in *Luggage* and *Person*.

them to avoid confusions with other small objects adjacent to the persons, as luggage, raised hands, and so on...

The Number of heads has proven to be very efficient to discriminate in a first step *Groups* of persons from *Luggage* or single *Person*. After that, the *Dispersedness* and the *Extent* are used to discriminate between *Luggage* and *Person* by using a K-NN classifier [18]. The Figure 5 shows the decision tree used in the current low-level classification. To use these two latter features (*Dispersedness* and *Extent*) in the K-NN classifier a previous normalization was required, in order to manage dimensionless numbers between 0 and 1. The *Extent* is already obtained normalized between 0 and 1, but the *Dispersedness* has been used after dividing it by 4π (theoretical value of the *Extent* of a circle) and using its inverse. That is, we use the normalized *Inverse Dispersedness (ID)* as: $ID = 4\pi / Dispersedness$.

Thus, *Luggage* blobs will normally exhibit high *ID* values (near to 1) while *Persons* will have lower values (near to zero). To illustrate the benefits of these features, in the Figures 6 and 7, the statistical distribution of their values for the blobs of the training database, are represented.

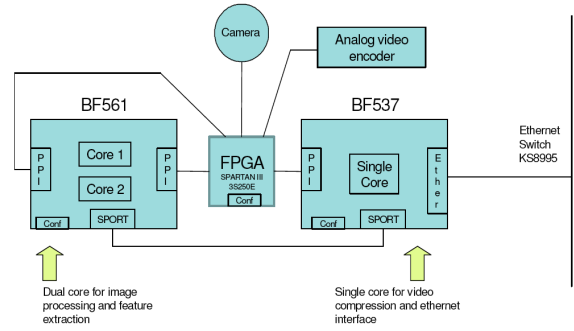


Figure 8. Video Board Hardware Outline

IV. VIDEO PROCESSING ARCHITECTURE AND IMPLEMENTATION

The video board has been developed using DSP processors for video processing tasks as well as a FPGA dedicated to image capture and sharing the images with the DSP processors [3], [19]. This cooperation between DSP and FPGAs has been also reported recently in different applications related to video processing [20], [21], [16], [22].

In Figure 8 the outline of the video board of the SENSE node is depicted. The actual implementation of the video board uses two Blackfin DSP processors. The idea is to have a dedicated dual core BF561 to process images and feature extraction. The features extracted will be sent to another DSP (BF537) through SPORT connection (no video is sent by this channel). The BF537, that has Ethernet interface, will be dedicated to video compression tasks as well as to external communication interface.

Blackfin core modules provides a PPI port to connect digital cameras following YUV, YCbCr and RGB standards. The camera programming can be done through I^2C interface using the Serial Camera Control Bus (SCCB) standard. The used camera in the current prototype is the Omnivision OV7660 (Colour, VGA resolution).

In order to provide flexibility to the design, all Blackfin PPI, camera output and analog video encoder are connected to a FPGA chip (SPARTAN III 3S250e). This configuration allows different connections among these buses. In particular, the FPGA is programmed to perform the following tasks:

- Video redirection: the camera output is connected to the FPGA and the program copies the image flow to FPGA outputs connected to BF537 and BF561 PPI ports.
- Local image time stamping.

The video board is charged with the tasks corresponding to the compression of live images and the extraction of characteristics of the video detected objects, as well as digital video capture. Some of these functions are provided directly by the hardware, and the rest are done by software running on both DSP processors. Mainly, these tasks are:

- Video compression and streaming (JPEG).
- Real-Time processing the images to identify objects
- XML encoding the image features and transmitting them to the main board (RDU) trough Ethernet (CIM module).

The FPGA interface is responsible for pin assignments through all the board. This approach enhances the versatil-

ity, making that some changes can be made into the hardware without affecting the software. It is also responsible of getting the frames (video capture) and delivering them to both cores. The result of video processing done in BF561 is transmitted to BF537 through a SPORT connection.

A. Image and features alignment

Object identification and video compressing services work in separate processors. As each processor has its own clock, we need a mechanism to align the XML messages with JPEG compressed frames. For this purpose, a 64-bit frame counter has been included into the board. This counter is incremented with each vertical sync signal coming from the camera. For each frame, the FPGA writes the time stamp provided by the counter in the first bytes of the image. As the processors share the frames, they will have a common time stamp for each frame. This is a simple way of getting a common time measure, and it does affect neither the detection of objects nor the video compressing algorithms.

V. LOW-LEVEL VIDEO SOFTWARE IMPLEMENTATION

A. Software for feature extraction unit (BF561)

The feature extraction unit is responsible for detecting objects in each frame and tracking them, creating a list of low level items with all the features found. The work can be divided into five main phases, as described in previous section: Acquisition, Segmentation, Labeling, Tracking and Classification

The BF561 processor has two cores. This means that at least two tasks can work concurrently. Tasks are assigned through the cores according to their computational weights. The algorithm as a whole must be executed in a sequential way, i.e. each frame goes through the five processing phases in order. Another important issue is that the acquisition is done via a Direct Memory Access (DMA) transfer. This means that it doesn't consume processor time, and that this task could work concurrently with others even into the same core. According to their CPU usage ratios, the tasks have been divided into three main working units, all of whom can work concurrently, so their performance is maximized. These working units are:

- Acquisition. Running on Core A.
- Segmentation. Running on Core A.
- Labeling, Tracking and Classification (LTC). Running on Core B.

The system runs as a software pipeline. Traditional pipelines are implemented as independent phases. Phases are interconnected by buffers. Each one reads data from the input buffer, performs some processing and writes the results into the output buffer. For the phases to be able to work concurrently, each buffer should maintain several data objects. The main advantage of this approach is its ease of implementation. But, with several phases and buffers, two main problems arise:

- *Restricted access*. Should the last phase need some data from the first buffer, the data may be lost.
- *High latency*. With different running times for each phase, there will be a bottleneck when the processing time of phase i is lower than the processing time of

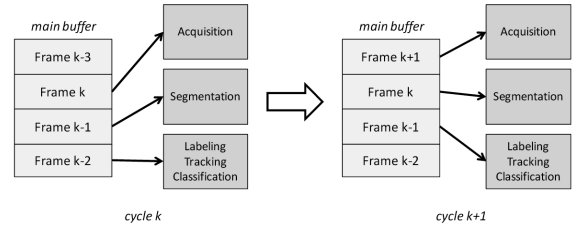


Figure 9. Software pipeline used to reduce the latency. Sequence shows cycles k and $k+1$. Note that *main buffer* is circular and in each cycle, one of the 4 frames remains unused.

phase $i + 1$. At this point, the communication buffer gets full, so the data has to go through longer queues and so the required time to cross the system increases.

Thus, based on the control logic of a segmented processor, a modified pipeline will aid to solve these problems. The new pipeline, based on a circular buffer (*main buffer*) is described in the Fig.9. This figure introduces the concept of system cycle. At each system cycle, each phase will work with a different frame. Each phase has two states: ready and working. When a new cycle begins, all phases go to working state. As each phase ends its work, it goes to ready state. When all phases reach the ready state, the system goes to the next cycle.

The control algorithm is implemented by means of events and event bits. An event is a binary event bit function. For instance:

$$E_1 = B_0 \wedge B_1 \wedge B_4 \quad (7)$$

defines event E_1 as a function of event bits B_0 , B_1 and B_4 . When all these bits are true, E_1 becomes true. In our system, each phase has an associated event bit. We have defined the event *nextcycle* as the conjunction of the event bits associated to each phase:

$$next\ cycle = B_{acquisition} \wedge B_{segmentation} \wedge B_{LTC} \quad (8)$$

The first task of all in video processing is to acquire the frames. As we have seen before in Figure 8, each processor gets the frames through the PPI parallel port. At this phase we have to:

- Get the frames
- Recover the time stamp included on the first line of the frames by the FPGA

Due to some restrictions, system initialization is done at the beginning of this phase: memory allocation, camera driver initialization and first background model creation. The frame acquisition is done via a DMA-PPI transfer. Data comes from the camera, through an 8-bit parallel port to the PPI device. This device works as a bridge from the camera to system memory. It must be taken into account that the first line of each image has some extra data: the time stamp. This data must be recovered and saved into the frame structure.

The frames are transferred in *YUV* format. For each single frame, we have 640×480 Y values, 320×480 U values and 320×480 V values.

B. Software for compression and Ethernet interface unit (BF537)

The BF537 processor is responsible of node communications with two main tasks:

- Compressing the incoming frames into JPEG format
- Generating the XML messages with the data obtained by the other processor.

All this data is sent through a Ethernet connection. The system has been designed to keep working even if there are no Ethernet connections available. Also, only one connection is allowed for each service. Each service is managed by means of a thread and a semaphore.

Compressed video is sent in JPEG frames. The first part developed has been the JPEG video service. Four threads manage the service, which has been developed as a pipeline with some special issues. The JPEG thread includes the time stamp generated by FPGA. This is done by modifying the header of the generated JPEG and inserting the time stamp.

The data obtained from the scene processing will be transferred using XML messages. The XML Stream service works receiving serialized scene data through the Serial Port and generating an XML ASCII message. This service is managed in a similar way as the JPEG.

VI. EXPERIMENTAL RESULTS

In this section, the measured results obtained from a first prototype of the node working in a real scenario are given as a demonstration of the possibilities offered by our video platform.

A. Time spent in each phase of video processing

The Figure 10 shows the time spent in the main phases of the video processing in the Core B of the BF561 processor during a short sequence taken from a scene with normal activity, as well as the total time between frames. As can be seen from this figure, the mean total time spent is about 133ms approx. This time is not casual, as it is due to the fact that the capture frame rate has been adjusted to 7.5 fps (that is: 133.33ms per frame) to avoid irregular framerate in overloads. From the data used to plot the Figure 10, we can obtain the following mean values for each phase (these values are orientative only, as the times are dependent on the complexity of the scene):

- *tracking*: 0.25ms
- *labelling*: 45.25 ms
- *classification*: 12.15 ms

B. Low-level classification results obtained from real prototype.

The previous section III described algorithms for segmenting scenes, tracking objects, low-level classifying objects and adapting the scene background to changes in light conditions. All these algorithms have been implemented and tested in a first development stage inside the Matlab environment. Afterwards, these algorithms have been embedded into the DSP processors.

Most of the experiments done are aimed to test the quality of the final classification of objects, as this is the goal of the complete process. The current results are satisfactory, but improvements are still possible and desirable.

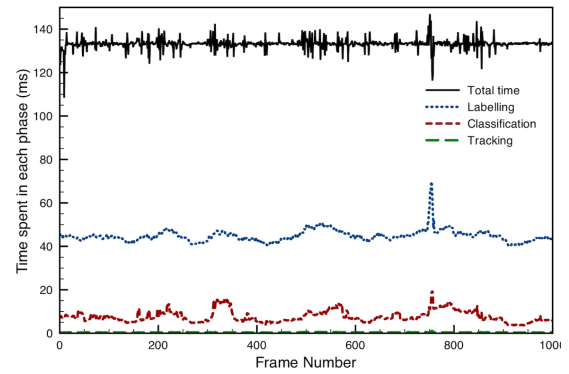


Figure 10. Time spent in each phase of the processing during a real execution in the node.

Table I
CLASSIFICATION RESULTS OBTAINED WITH THE IMPLEMENTED SOFTWARE RUNNING IN THE PROTOTYPE DURING A REAL EXPERIMENT.

Object Class:	Classified as:		
	Person	Group	Luggage
Person	91.9%	19.3%	0.3%
Group	6.8%	80.7%	0.5%
Luggage	1.3%	0%	99.2%

Thus we continue working on these issues, enhancing the current silhouettes databases and obtaining more videos with ground truth, to make better evaluation tests.

To evaluate the real behaviour of the prototype, a short real sequence was taken in our School main hall, with about 3000 frames. The obtained classification results as well as the images have been recorded and the classification results have been inspected frame-to-frame in a manual way. From this short scene, a total of 643 persons, 202 groups and 390 luggages were detected and classified by the prototype. The classifications results obtained have been summarised in the Table I.

A client application connected directly to the video processing module (see Figure 11) has been used for “black box” testing purposes. The preliminary tests carried out on this prototype show satisfactory results in the full video processing, achieving frame rates up to 8 frames per second including JPG video images as well as XML streaming. Also, we expect that future work will improve the current achieved results.

VII. CONCLUSIONS

In this paper, the details of the implementation of the video algorithms in the SENSE node have been described, including the theoretical foundations and the experiments carried out to evaluate the results of the elected algorithms that have finally been implemented in the video board. These algorithms include: algorithms for segmenting scenes, tracking objects, low-level classifying objects and adapting the scene background to changes in light conditions. All these algorithms have been implemented and tested in a first development stage inside the Matlab environment. After that, these algorithms have been embedded into the DSP processors.

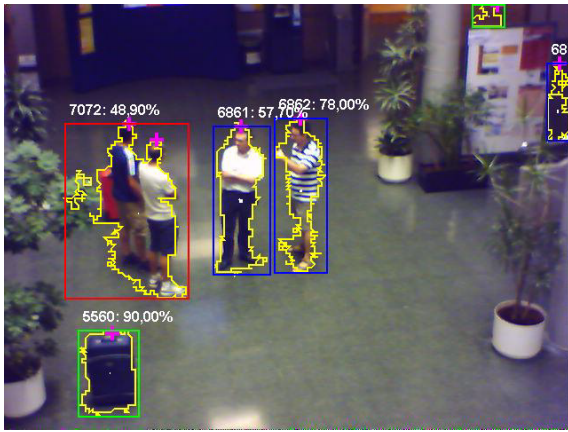


Figure 11. Image obtained from a client application showing information provided by low level video processing services. Each detected object has a label with the object identifier number and the assigned membership probability. Each BoundingBox is drawn with a color that represents the class, and the detected heads have been marked with a cross.

Classification results, as being the output of the process, may be measured as a whole, considering the total amount of objects expected to be segmented, tracked and classified as a single stage. The results obtained in our real experiments seemed very satisfactory, as expected from our previous off-line experiments with Matlab environment. Table I summarizes the classification results obtained. In any case, for all tested videos, the True Positive rate in all classes is higher than 80%, being this the limit agreed to ensure high level robustness.

REFERENCES

- [1] L. Fuentes and S. Velastin, "From tracking to advanced surveillance," in *Proceedings of International Conference on Image Processing Conference (ICIP 2003)*, vol. 3, pp. 121–124.
- [2] N. Nguyen, S. Venkatesh, G. West, H. Bui, and A. Perth, "Multiple camera coordination in a surveillance system," *Acta Automatica Sinica*, vol. 29, no. 3, pp. 408–422, 2003.
- [3] S. Hengstler and H. Aghajan, "Application-oriented design of smart camera networks," in *2007 First ACM/IEEE International Conference on Distributed Smart Cameras*. IEEE, 2007, pp. 9–16.
- [4] G. Foresti, C. Micheloni, L. Snidaro, P. Remagnino, and T. Ellis, "Active video-based surveillance system: the low-level image and video processing techniques needed for implementation," *IEEE Signal Processing Magazine*, vol. 22, no. 2, pp. 25–37, 2005.
- [5] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 34, no. 3, pp. 334–352, 2004.
- [6] I. Haritaoglu, D. Harwood, and L. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 809–830, 2000.
- [7] J. Rosell-Ortega, G. Andreu-Garcia, A. Rodas-Jorda, V. Atienza-Vanacloig, and J. Valiente-González, "Feature Sets for People and Luggage Recognition in Airport Surveillance Under Real-Time Constraints," in *Proceedings of VISIGRAPP08*, 2008, pp. 662–665.
- [8] S. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler, "Tracking groups of people," *Computer Vision and Image Understanding*, vol. 80, no. 1, pp. 42–56, 2000.
- [9] C. Sacchi, G. Gera, L. Marcenaro, and C. Regazzoni, "Advanced image-processing tools for counting people in tourist site-monitoring applications," *Signal Processing*, vol. 81, no. 5, pp. 1017–1040, 2001.
- [10] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.
- [11] SENSE, "Smart embedded network of sensing entities. <http://www.sense-ist.org/index.html>."
- [12] M. Piccardi, "Background subtraction techniques: a review," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, 2004, pp. 3099–3104.
- [13] H. Grabner, P. Roth, M. Grabner, and H. Bischof, "Autonomous learning of a robust background model for change detection," in *Proc. PETS*, 2006, pp. 39–46.
- [14] B. Shoushtarian and H. Bez, "A practical adaptive approach for dynamic background subtraction using an invariant colour model and object tracking," *Pattern Recognition Letters*, vol. 26, no. 26, pp. 5–26, 2005.
- [15] F. Chang, C. Chen, and C. Lu, "A linear-time component-labeling algorithm using contour tracing technique," *Computer Vision and Image Understanding*, vol. 93, no. 2, pp. 206–220, 2004.
- [16] J. Garcia, O. Perez, A. Berlanga, and J. M. Molina, "Video tracking system optimization using evolution strategies," *International Journal of Imaging Systems and Technology*, vol. 17, no. 2, pp. 75–90, 2007.
- [17] A. Lipton, H. Fujiyoshi, and R. Patil, "Moving target classification and tracking from real-time video," in *IEEE Workshop on Applications of Computer Vision*, vol. 14. IEEE, 1998, p. 3.
- [18] P. Devijver and J. Kittler, *Pattern recognition: A statistical approach*. Prentice Hall, 1982.
- [19] M. Pradhan, "Simplified micro-controller & FPGA platform for DSP applications," in *2005 IEEE International Conference on Microelectronic Systems Education, Proceedings*. IEEE Computer Soc., 2005, pp. 87–88.
- [20] W. Jun, L. Peng, and L. Wei, "Design and implementation of a real-time image processing system with modularization and extensibility," in *2008 International Conference on Audio, Language and Image Processing, Vols 1 and 2, Proceedings*. IEEE, 2008, pp. 798–802.
- [21] H. Khali, M. Riyah, and A. Araar, "Real-time 3d image computation using lut-based dsp systems," in *The 2004 IEEE Asia-Pacific Conference on Circuits and Systems*. IEEE, December 2004, pp. 369–372.
- [22] J. Gao, H. Tao, and Y. Wen, "The design and implementation of a high-speed video image collecting system based on DSP technology," in *ISTM/2005: 6th International Symposium on Test and Measurement, Vols 1-9, Conference Proceedings*. International Academic Publishers Ltd., 2005, pp. 6460–6464.