

Sensor Data Fusion for Activity Monitoring in Ambient Assisted Living Environments

M. Amoretti^{1,5}, F. Wientapper², F. Furfari³, S. Lenzi³, and S. Chessa^{3,4}

¹ R&S INFO, Parma, Italy,

`michele.amoretti@unipr.it`

² Fraunhofer IGD, Darmstadt, Germany,

`folker.wientapper@igd.fraunhofer.de`

³ CNR-ISTI, Pisa, Italy,

`{furfari,lenzi,chessa}@isti.cnr.it`

⁴ Computer Science Dep., Univ. of Pisa, Italy

⁵ Dep. of Information Engineering, Univ. of Parma, Italy

Abstract. We illustrate the PERSONA context-awareness framework applied to a major problem in Ambient Intelligence, namely user activity monitoring, that requires to infer new knowledge from collected and fused sensor data, dealing with highly dynamic environments where devices continuously change their availability and (or) physical location. We describe the Sensor Abstraction and Integration Layer (SAIL), we introduce the Human Posture Classification component, which is one particular context information provider, and finally we describe the Activity Monitor, which is a reasoner that delivers aggregated/derived context events in terms of the context ontology.

Key words: ambient intelligence, context-awareness, data fusion, artificial vision, activity monitoring

1 Introduction

The concept of Ambient Intelligence (AmI), which refers to a digital environment that proactively supports people in their daily lives, was introduced by the Information Society Technologies Advisory Group (ISTAG) of the European Commission [13]. AmI overlaps with other concepts, such as ubiquitous computing, pervasive computing, context awareness, embedded systems and artificial intelligence [7].

In the AmI context, the European Commission recently started the Ambient Assisted Living (AAL) technology and innovation funding programme, aiming at improving the quality of life of older people in their homes, by increasing their autonomy and assisting them in their daily activities, and by letting them feeling included, secure, protected and supported. AAL spaces are physical places featured with AmI enabling technologies, including the intelligence which supports the services. Examples of AAL spaces are the home where the user lives, the neighborhood, the town, but also the body of the user itself. The technical challenge is to develop an integrated technological platform that allows the practical

implementation of the AAL concept for the seamless and natural access to those services indicated above, to empower the citizen to adopt ambient intelligence as a natural environment in which to live.

For the creation of AmI environments, it is essential to have a framework supporting context-awareness. The scope of the contextual information spans the situational user context (his or her identity, capabilities, preferences, tasks, and state) over temporal, spatial, and environmental parameters (*e.g.* time, location, temperature, etc.), the available and accessible resources, and their capabilities and states. In general, context sources are manifold (several sensors, different profiles, and still others not enumerated explicitly), and the data coming from these sources must comply with a shared data model (the context ontology) in order for them to be used further by the consumers. In particular, some multidimensional (*i.e.* non-binary) sensors such as cameras, microphones, inertial sensors, must have a conversion of their data (pixel values, bit-encoded acoustic signals, etc.) into basic but at least meaningful, ontological states. Moreover, there must be an aggregation and reasoning mechanism aimed at deriving more significant context information, and there must be a subscription and notification mechanism aimed at triggering actions within an AmI system.

In this paper we illustrate a modular solution for sensor data fusion, allowing run-time connection and disconnection of components (sensors, data filters, reasoners, actuators). This is made possible by the PERSONA context awareness framework (introduced in section 2), which defines the concept of *context bus*, shared by specialized software components that produce and consume context events characterized by different granularity. Some components are directly connected to hardware sensors, and publish raw data in the context bus. Such basic context events are consumed by specialized data filters and reasoners, that infer high-level knowledge. The PERSONA middleware enables data fusion and aggregation at three levels: network, virtual network and application.

Such a modular approach allows to cope with a wide range of context awareness problems. In this paper we focus on user activity monitoring, that needs to infer new knowledge from collected and fused sensor data, dealing with highly dynamic environments where devices continuously change their availability and (or) physical location (*e.g.* those which are carried or worn by the user). In Section 3 we describe the Sensor Abstraction and Integration Layer (SAIL), with a specific discussion on the integration of ZigBee devices. Then, in Section 4, we introduce one particular context information provider, the Human Posture Classification component, which processes images coming from a camera in order to retrieve static estimates about meaningful, ontological posture states of a person being in the field of view. Finally, in Section 5, we describe the Activity Monitor, which is a reasoner that delivers aggregated/derived context events in terms of the context ontology (in particular, the action ontology). In the final section we summarize the work presented in this paper and discuss future work.

2 The PERSONA framework for context awareness

PERSONA (Perceptive Spaces prOmoting iNdependent Aging) [2] is a EU-funded research project (FP6) started in 2007, aiming at developing a scalable open standard technological platform to build a broad range of Ambient Assisted Living (AAL) Services. In this context the main technical challenge is the design of a self-organizing middleware infrastructure allowing the extensibility of component/device ensembles in an ad hoc fashion. In order to achieve this goal the communication patterns of the infrastructure must be able to execute distributed coordination strategies in order to provide the necessary service discovery, service orchestration and service adaptation functionalities.

The components of a PERSONA system are interfaced with the PERSONA middleware that enables the allocation of a different number of communication buses, each of them adopting specific and open communication strategies. Components linked with the PERSONA middleware may register with some of these communication buses, find each others and collaborate through the local instances of the buses. Figure 1 shows the conceptual architecture of PERSONA. Input and output buses support multi-modal user interactions with the system. The context bus is an event-based channel to which context sources are attached, in particular the Wireless Sensor Networks (WSN) [5] are attached to this bus. Published events may be re-elaborated and transformed in high level events (situations) by components that have subscribed to the bus (*e.g.* context reasoners).

The service bus is used to group all the services available in the AAL-space, being them atomic or composite (whose availability is managed by a Service Orchestrator component). Services belonging to the service bus may be requested by the Situation Manager in consequence of situation detections and rules stored in the Knowledge Base of the system. Generally, devices are attached to both the context and service bus. The former is used to send notifications of status changes, the latter to answer to status query or execute actions (*e.g.* switch on the light device). Many other basic components are foreseen in PERSONA system, but their discussion is out of the scope of this paper (see [11] for details).

As in other international research projects (Gator Tech [17], Amigo [4], Socam [14]), we developed a middleware implementation based on the OSGi Platform [1]. The OSGi specification provides a standardized way to manage the software life cycle of Java applications. OSGi implementations are containers running on top of a JVM which enable installation, removal, start, and stop of components at run time. An OSGi component is a JAR file called *bundle* which contains Java classes, resources and metadata describing the dependencies with other bundles. A feature of the OSGi platform that is useful to recall here is the OSGi service model. A component can register with the OSGi Registry instances of services implementing specific interfaces and described by key-value properties. Every other component interested to monitor the presence of a particular service can register listeners with filter properties that will be notified by the OSGi framework when there is a match.

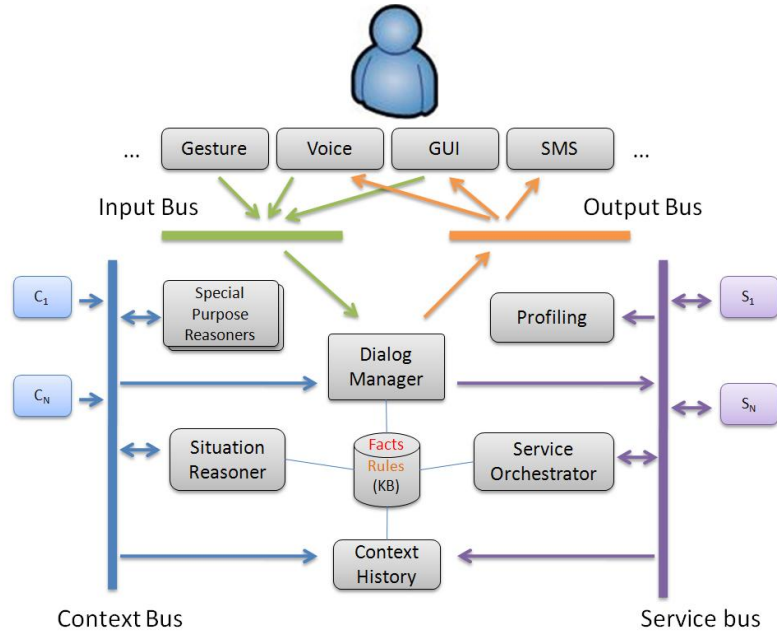


Fig. 1. The PERSONA Abstract Architecture

3 Sensor Abstraction and Integration Layer

The artifacts composing AAL-spaces can be classified in *stationary*, *portable* and *wearable components*. The first ones run on desktop PCs, Set Top Boxes, Residential Gateways or they belong to the environmental infrastructures like Home Automation sub-systems; *portable components* are, for instance, medical devices or mobile/smart phones, while *wearable components* are based on garments equipped with sensors. The PERSONA middleware targets small but reasonably powerful devices, therefore not all the components can be integrated by using an instance of the PERSONA middleware. Typically, wearable components, as well as nodes of WSNs or Home Automation Systems (HAS) require a different approach because of their limited computation resources; in such cases PERSONA adopts a solution based on a gateway, which allows the PERSONA application layer to share information by communicating with other application layers resident on different network infrastructures (*e.g.* ZigBee [5] or Bluetooth applications). Due to the features of the OSGi Platform¹, it is sufficient to write a bridging component that interacts from one side with the PERSONA middleware and from the other side with the software drivers that access the networked devices. In the following, we briefly introduce the Sensors Abstraction and Integration Layer (SAIL) developed in PERSONA, by highlighting different aspects

¹ Initially OSGi was designed as a framework to develop Open Service Gateways.

and requirements of sensor network applications. A specific section related to the integration of ZigBee devices concludes the chapter.

3.1 Wireless Sensor Networks

Wireless Sensor Networks are an important technological support for smart environments and ambient assisted applications. Up to now, most applications are based on ad-hoc solutions for WSNs, and solutions providing uniform and reusable applications are still in their youth. Note that AAL spaces can be populated by many sensor networks (*e.g.* ZigBee or IEEE 802.15.4 standard, and Bluetooth). The main requirements, in the integration of WSN into PERSONA, are:

- R1: *Integration of the different sensor network technologies.*
- R2: *Sharing of communication medium by concurrent sensor applications.*
- R3: *Management of different applications on the same WSN.*
- R5: *Dynamic discovery of sensor applications.*
- R6: *Management of logical sensors.*
- R7: *Configuration and calibration of sensor applications.*

The requirement R1 is usually satisfied by enabling the dynamic deployment of ad-hoc network drivers in the system, while we can assume that requirements from R2 to R5 largely depend on the operating systems and middleware used for the sensor nodes programming (*e.g.* ZigBee, TinyOS, TinyDB, TeenyLIME and others [22]). The requirement R6 concerns the possible virtualization of the sensors deployed either in the environment or worn by the users. For example, let's consider an application that detects the user's posture by means of a number of accelerometers placed on the body: in this case, the posture can be represented by a single logical sensor, which aggregates information produced by all the accelerometers for producing the state of the user (sitting, walking, etc.). In this example, the accelerometers may not even be visible to the application layer. Such a virtualization can be implemented directly on the sensor network or implemented at application level within the gateway.

In order to face these requirements, we developed SAIL [12] an architecture organized in three layers, namely the *Access*, *Abstraction*, and *Integration layer*, constructed over the OSGi platform. The Access layer is a collection of drivers that provides basic access to WSNs. The Abstraction layer, if needed, can be used to elaborate the data received by the sensors before of exporting them at application level, and the last layer exposes the sensors to one or more access technologies, that in the context of PERSONA project is the middleware.

3.2 ZigBee Networks Integration

A diagram of the current SAIL architecture with ZigBee networks is depicted in figure 2. The ZigBee Base Driver is a network driver in charge of executing

the scan of the network, getting the description of the ZigBee devices, and registering a proxy service for accessing the discovered remote services. This proxy is a generic ZigBee service, registered with the OSGi Platform, which exposes the properties retrieved during the network inquiry. It enables the access to the remote service by means of simple primitives (*e.g.* `invoke(byte[] payload)`) that have to be filled by the proper cluster message. The components on the upper layers may act as Refinement Drivers (in OSGi terms). By using the OSGi Framework facilities, as soon as a service implementing a known cluster is registered, these drivers refine the service by registering another service proxy with a more detailed interface (*e.g.* action/command based). Thus the second layer is specialized to represent the service according to a specific profile, for instance the Home Automation profile. The upper layer integrates the ZigBee services within PERSONA. It is composed of Sensor Technology Exporters (STE), which discover the services by implementing standard or extended profiles and registers proxies that are PERSONA-aware components². The mapping between services compliant to the ZigBee model and PERSONA model is realized at this level; these proxies send events or register service callees according to the PERSONA ontologies.

In conclusion, the abstract layer is populated by custom drivers which may combine and process the sensed data to instantiate logical sensor services and refine the cluster-based services. All the code for ZigBee network access will be released with OS license as soon as the integration with a commissioning tool will be completed.

4 Vision-based human posture classification

Concerning the *acquisition* of context information, some sensors require a transformation of their output into meaningful, ontological states such that reasoning mechanisms may work upon it. In this section we will introduce one particular context information provider that has been developed in the course of the PERSONA project. The Human Posture Classification (HPC) component is a camera based solution designed to retrieve static estimates about discrete (*i.e.* ontological) posture states of a person being in the field of view of the camera. "Static" means that the postures are derived from single images and are related to one concrete time instant. Furthermore, in contrast to the main research stream on human pose estimation (see [21] and [25] for in-depth reviews) no continuous pose is estimated. So instead of estimating the configuration of the skeletal configuration of the articulated body in a parameterized fashion *e.g.* by means of expressing it by joint angles of limbs and the general body orientation, the postures are directly classified according to a finite set of discrete posture states such as "Sitting", "Standing", "Bending", etc. Such a representation is needed to support higher level reasoning within a semantic / ontological AAL-architecture like the PERSONA framework.

² The same approach can be used to expose the services according other access technologies (*e.g.* UPnP).

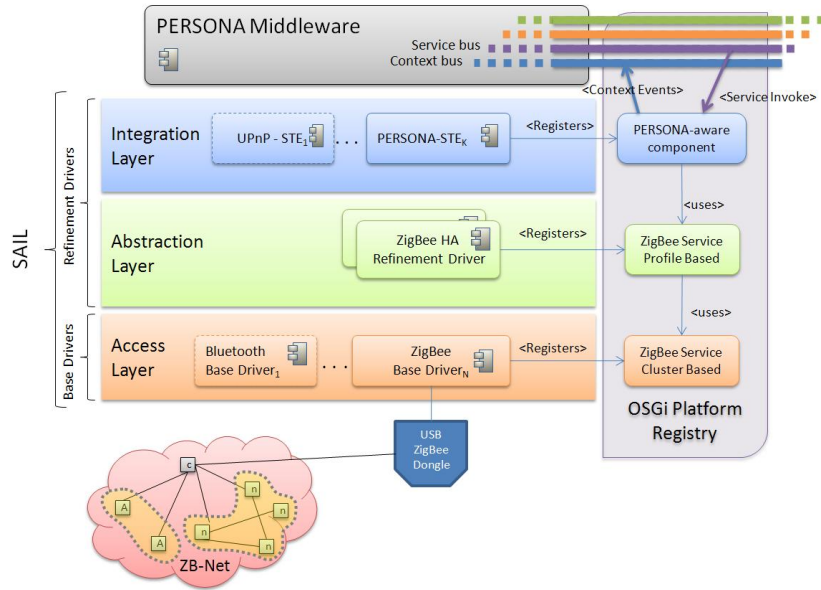


Fig. 2. The SAIL layered architecture

Compared to related work in the field of discrete posture classification for domestic services ([10], [9], [23]), our approach is characterized by its simplicity and effectiveness. Firstly, we do not require multiple cameras in order to resolve ambiguities arising from self-occlusion of the body parts or to handle difficulties caused by poor segmentation when using conventional color based cameras. Instead, we advocate the use of Time-Of-Flight cameras which produce a three-dimensional, geometric representation of the scene. This reveals much better segmentation results and provides additional information for ambiguity handling. Secondly, we do not use a particular model of the human body in order to estimate a complete, parameterized body configuration prior classification. In fact, only few image processing steps are needed to obtain a low resolution feature image which in turn is used to train a recently proposed machine learning approach. It is clear that due to the high amount of computational effort needed for image processing applications, the HPC-component should run on a dedicated machine. It is the output of this preprocessing step - the extracted context information (posture states) - which is fused within the communication framework for Activity Monitoring.

In the following two subsections a brief overview over the particular advantages of using TOF-cameras instead of conventional types is given. Next, we explain the image processing steps for obtaining feature vectors that are used for training and online classification. Technical details and an evaluation of the solution are presented in [30].

4.1 Advantages of Time-Of-Flight cameras

Common to most of the human pose estimation approaches is the assumption that the camera is placed at a fixed spot in the room. This allows for applying a background-foreground segmentation (or background-subtraction) technique to obtain a silhouette image of the person. The basic idea is that the appearance of the scene is learned once beforehand, without any person in the field of view of the camera. While processing the live image the foreground can in turn be segmented by identifying changes with respect to the background model.

In fact, most pose estimation approaches rely heavily on a good segmentation result. However, despite many years of research (see [24] for a review on background segmentation), in practical, realistic environments conventional cameras still pose certain difficulties to obtaining a clear silhouette. These difficulties include the following:

- noise in the image,
- similar colors of the background and the clothes worn by persons,
- shadows that result in additional, unwanted variations between the background model and the live images,
- changing lighting conditions (*e.g.* when someone switches a light on or off) and,
- completely dark environments, as in the case of monitoring the behaviour of people at night, where conventional cameras are not applicable, at all.

Due to these difficulties we propose to use images coming from Time-Of-Flight (TOF) cameras ([20]), instead. TOF cameras produce a three dimensional, geometrical representation of the scene by emitting infra-red light and measuring the time the light takes to returning back to the camera.³ With TOF-cameras most of the abovementioned difficulties are overcome, as the representation of the scene is based on its geometry and not on its visual appearance, thus making it more robust and more applicable in realistic scenarios.

4.2 Time-Of-Flight image feature extraction, learning and classification

The implemented algorithmic solution to camera based posture classification comprises four main steps which are briefly summarized, subsequently:

1. *Background subtraction and connected component merging*: Although TOF-cameras directly produce a cloud of 3D-coordinates of the perceived scene as output, each depth value is also associated to a pixel-coordinate. Thus many of the conventional image processing techniques may be applied to TOF-images in a similar fashion, including background-subtraction. However, compared to grey-level images from conventional cameras, the pixel

³ More precisely, a TOF camera emits sinusoidal light impulses and measures the phase shift of the returning light for each pixel.

values (depth-value) have much higher and very heterogeneous noise. To account to these peculiarities we adopted a recently proposed algorithm [26] that estimates the mean and covariance for each pixel in the background model (see figure 3, top-right). Next, the perceived differences between the background and the live image are grouped into connected pixel regions ("blobs"). This step is used to remove any false positives arising *e.g.* due to image noise. A simple threshold is applied to the size of the connected components, *i.e.* blobs below a certain size are simply discarded. Furthermore the largest blob among the remaining is assumed to be the one corresponding to a person.

2. *Subimage cropping, resampling and vectorization*: Based on the previous region-of-interest detection, a sub-image is cropped in such a way that it contains as much foreground as possible. The sub-image is centred and resampled to a low resolution while keeping the aspect ratio constant (see figure 3, bottom-left). The cropped and resampled sub-image is interpreted as a high dimensional feature vector with each pixel value being one element of the vector (*e.g.* a 20-by-24 image results in a 480 dimensional vector).
3. *Learning a transformation from image space to posture space (offline)*: In order to infer the posture from the high dimensional feature vectors, a recently proposed machine learning approach is used ("Locality Preserving Projections") [15]⁴. A set of reference images needs to be captured and annotated manually according to their corresponding discrete states ("sitting", "standing", etc.). The machine learning approach is closely related to Principal Component Analysis (PCA), but, by contrast, takes into account neighbouring relations between the feature vectors, and thus, can also be used in a supervised manner (refer to [27] for details). It works by finding an optimal transformation from high dimensional feature space to a low dimensional representation. Image vectors belonging to the same or similar state are mapped to nearby points in the low dimensional space, such that points belonging to same states form clusters (see figure 3, bottom-right). For each cluster the mean and covariance is computed, which are used later for classifying the transformed image vectors into the corresponding states.
4. *Feature projection and classification (online)*: The optimal projection derived from the training data is applied to the online image vectors. This projected online feature is then classified according to the closest mean vector of the projected training data. Furthermore we apply a threshold on the Mahalanobis distance between the online feature vector and the covariance of the projected training data of each class (see figure 3, bottom-right). *I.e.* if the normalized distance to the closest mean exceeds a certain threshold, the state is considered to be "Unknown". As we can show on real data, the machine learning approach has good generalization capabilities, *i.e.* previ-

⁴ Wang and Suter ([29], [28]) already adopted LPP in the context of human monitoring. However their approach differs from ours in that they use it to obtain a parameterization of the image manifold and, in turn, use the evolution of the projected feature vectors to classify dynamic activities

ously unseen images not contained in the training set are correctly projected and classified.

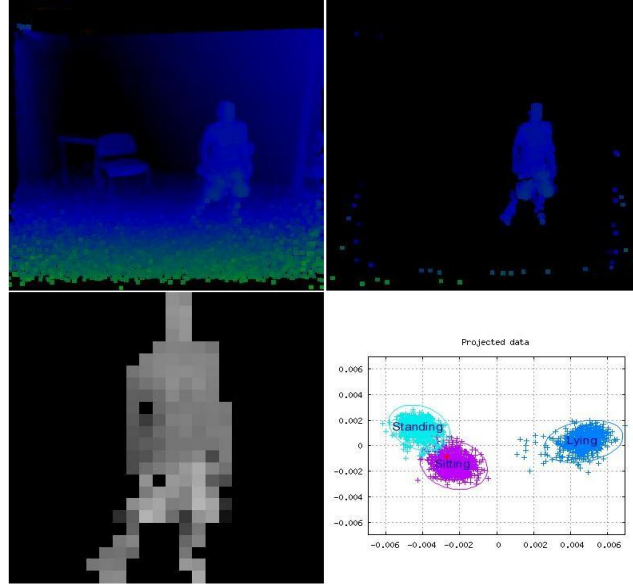


Fig. 3. Illustration of the image processing and classification steps for the human posture recognition (best viewed in color): Color encoded visualization of the 3D raw data coming from a TOF-camera (tl), result after background-subtraction (tr), cropped feature-image with grey-valued encoded relative depths (bl), projected training data for three states, "Standing", "Sitting" and "Lying" (cyan, purple and blue), and the projection of the current feature image on the left (red point above the "Sitting" caption)(br).

5 Data fusion for user activity monitoring

Understanding of human behaviors may be thought as the classification of time varying feature data, *i.e.* matching an unlabeled set of context events and other useful information with a group of labeled reference sets representing typical behaviors. A distinction between *static* and *dynamic* activities is necessary. Static activities like "standing" or "sitting" can be inferred directly from low-level data at a particular time instant (such as the pose of the person at a certain time using some kind of thresholding mechanism on the pose estimate). By contrast dynamic activities, such as "moving around", are usually composite activities requiring a monitoring of a full sequence of low-level data (*e.g.* context events describing ongoing sub-activities). Low-level data needs to be stored for several

time frames (in a context buffer), as the whole sequence is needed to infer that activity from an evolution of the low level data. For example: "cooking" may be composed of several low-level data at different time instances: "opening the fridge", "closing the fridge", "standing in front of the oven", etc.

The existing body of literature in automatic reasoning can be decomposed according to two basic approaches. A pure "rule-based" approach that is a composition of several "If...Then..." conditions. *E.g.* for "cooking", it might be used "fridge has been opened for several times" && "hotplate was on for > xx minutes" && "activity of person in the kitchen" && "...". A "supervised learning" based approach [18], where the system is trained with a history of low-level data and corresponding output states (the low-level data is labeled with corresponding output states). Usually such supervised learning methods work by finding a decision surface in the high-dimensional (labeled) input data. Supervised learning (a.k.a. inductive machine learning) applied to human behavior detection has two advantages. First, the system can be configured (trained) by performing the activities themselves instead of programming the rule-based conditions. Second, the system can be updated/trained online, for example by asking the person if the inferred activity was correct. Then with the confirmation/disagreement of the user, the training space (*e.g.* decision surface) is updated.

Our approach is based on probabilistic rules, with absolute probabilities of high-level actions can be automatically updated by means of long-term behavior detection (which is not discussed in this paper). The Activity Monitor (AM) is a reasoner that delivers aggregated/derived context events in terms of the context ontology (in particular, the action ontology). Thus the AM registers to the context bus (c-bus) both as subscriber and publisher in order to use context events from lower levels, and produces context events on a higher level. In details, the AM is a special-purpose reasoner that aggregates context events related to basic user activities (**AtomicActions**), recognized by the HPC component, and to user localization, as well as information about the status of user-controlled devices, such as the TV, to produce context events describing complex user activities such as "eating", "watching tv", etc (**CompositeActions**).

Figure 4 summarizes the internal architecture of the AM, which is registered on the c-bus as context subscriber for many user-related context events (*e.g.* those published by the HPE). When a context event of type **AtomicAction** is received, the Classifier decides whether to consume it immediately (to infer a **CompositeAction** and publish it as context event) or to store it in the Action Archiver, for later use.

The Classifier is based on a Bayesian network (BN) approach [16, 19]. A Bayesian network is a graphical model enabling the description of probability relationships among a set of variables (features). The graph allows the user to understand which variables affect which other ones, and also serves as the backbone for efficiently computing marginal and conditional probabilities that may be required for inference and learning. Because a Bayesian network is a complete model for the variables and their relationships, it can be used to answer probabilistic queries about them. For example, the network can be used to find

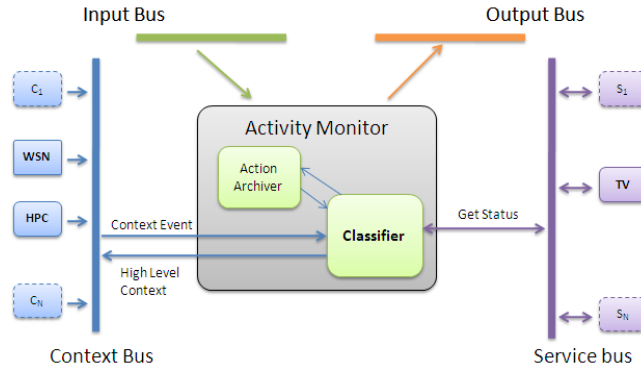


Fig. 4. The internal architecture of the Activity Monitor.

out updated knowledge of the state of a subset of variables when other variables (the evidence variables) are observed. The process of computing the posterior distribution of variables given evidence is called probabilistic inference.

We illustrate how BNs have been used to implement the Activity Monitor with a simple example. Then we briefly illustrate the classification algorithm implemented in the AM.

5.1 Case study

Different context events may allow to infer that the user is involved in a telecommunication: he is talking, he is facing TV, the TV is switched on in "communication mode". Being in the living room, with TV switched on may also mean that the user is watching TV programmes. From the point of view of the Activity Monitor, user actions (A_i) and localizations (L_j), as well as and device states obtained from the s-bus (S_k) are boolean random variables. The idea of inferring situational knowledge from a set of answers to "Y/N" questions comes directly from Shannon's information theory. In this context, asking more questions means merging more c-bus and s-bus data, that allows to infer more detailed knowledge.

The BN in figure 5 illustrates the dependencies among the variables considered for the case study. Those representing composite actions are placed as roots in the acyclic graph, while sub-actions and device states are always leaf nodes (since they are considered to be implied by composite actions).

The BN designer has to fill the conditional probability tables (CPTs) for each variable. For example, when a "Telecommunicating" action is ongoing, the probability that the user is in the living room is 0.5, otherwise it is 0.2. These values can be set taking into account the average time spent by the user, each day, to perform the composite actions of interest (cooking, watching TV, etc.), observed by relatives, caregivers and friends. Their values can be updated automatically, by using long-term behavior detection.

The Activity Monitor is thus able to compute, for example:

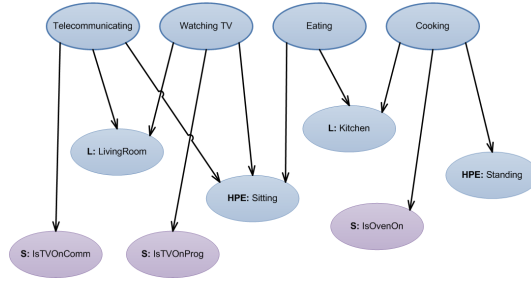


Fig. 5. Bayesian network for the case study.

$$P(\text{"Telecommunicating"} \mid \text{"L:LivingRoom"}=T, \text{"S:IsTVOnComm"}=T)$$

i.e. "what is the likelihood that the user is telecommunicating, given that he is in the living room, and the TV is in communication mode?"

5.2 Implementation of the Activity Monitor

To implement the classifier based on BNs, we adopted the Weka library (version 3.5.8) [3, 6], which is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from Java code. For Bayesian network design and management, Weka provides a Java API and a graphical editor.

The Weka BN GUI allows to create Bayesian networks and to store them on files in XML BIF format [8]. To create a BN, nodes can be placed and linked together in a free space, and CPTs can be defined. A BN stored in a XML BIF file can be read, modified and stored in a new file by means of the Weka API.

Context events describing simple actions (such as "Sitting") are used by the AM to infer the **CompositeAction** context events. If a simple action does not contribute immediately to the inference of a composite action, it is stored in the Action Archiver, which is a continuously updated list.

When a context event describing a basic **Action** is intercepted, the BN is explored and the composite action with higher marginal probability is returned, using the following algorithm:

1. get node i from the BN, such that $(\text{descr}(i) == \text{descr}(\text{theAtomicAction}))$
2. get the set N_i of parents of node i
3. for each node in N_i compute the marginal probability P_m , considering other evidences
4. select node j in N_i with the highest marginal probability $P_m(j)$
5. return $\text{descr}(j)$ and $P_m(j)$

Step 4 is the most important. It requires to take into account, for each node in N_i , the current values (evidences), whether they are available, of all its child nodes (that do not always represent context events, but information to be requested on the s-bus). For example (still referring to the case study), suppose the

"Sitting" event has been detected. Its parent set N_i includes "Watching TV", which has two more child nodes, one describing the presence of the user in the living room, the other describing the programming state of the TV. By invoking an appropriate service, the AM can retrieve the current value of the variable represented by the "S:IsTVOnProg" node (T or F). Moreover, the AM can check its internal location archive, searching for recent evidence of use presence in the living room. These facts are then used to compute the marginal probability of the considered parent node, *e.g.*

$$P(\text{"Watching TV"} \mid \text{"Sitting"}=T, \text{"L:LivingRoom"}=T, \text{"S:IsTVOnProg"}=F)$$

If a parent node has, among its child nodes, other context events representing actions, the AM inspects the Action Archiver, looking for recent evidence of such actions. If no recent evidence of actions related to a composite action is found in the archive, the evidence of those actions is set to F.

6 Conclusions

The PERSONA framework that we presented in this paper addresses three major issues for user related context awareness in an AmI environment.

We presented a general *communication framework* that allows to exchange context information, thus supporting context reasoning and data fusion in a distributed manner. A key concept is the context bus, to different components may subscribe for publishing and consuming context events. Within this paper a special focus has been given to the design of the wireless sensor network integration. The framework supports data fusion and aggregation at three levels of abstraction: network, virtual network (SAIL) and application level.

We addressed the problem of *context data acquisition*, *i.e.* the preprocessing of sensor output in order to obtain meaningful, ontological information such that high-level reasoners may work upon it. Here we focused on a particular problem in computer vision: the acquisition of posture information of the user when captured by a camera. We suggested to use Time-Of-Flight cameras to circumvent several difficulties (*e.g.* changing lighting conditions) that usually decrease the robustness of any solution in realistic and practical situations. Furthermore, we adopted a machine learning approach that can be trained to detect a variety of different posture states of interest. Of course, postures of persons are of special interest for activity monitoring as they provide important clues about the activities being performed.

Finally, a powerful and flexible *context data fusion* approach was introduced, *i.e.* the Activity Monitor. It is a reactive component that produces information about ongoing user activities, merging context events such as human postures and localization, with information about the status of user-controlled devices. The Activity Monitor is composed by a classifier and a short-term context archiver. The classifier is based on a Bayesian network (BN) approach, which allows to infer high-level knowledge with associated marginal probability. To

improve the component, we are studying mechanisms for allowing to dynamically update the absolute probabilities of composite actions, taking into account the results produced by another PERSONA reasoner which is the Long-term Behavioral Analyzer (whose development is in progress).

Acknowledgements

This work has been partially funded by the EU IST Project PERSONA (FP6 contract N.045459).

References

1. OSGi web site: <http://www.osgi.org>.
2. PERSONA project web site: <http://www.aal-persona.org>.
3. Weka 3: Data mining software in java. <http://www.cs.waikato.ac.nz/ml/weka/>.
4. AMIGO project. Ambient intelligence for the networked home environment: www.amigo-project.org, September 2004.
5. P. Baronti, P. Pillai, V. W. C. Chook, S. Chessa, A. Gotta, and Y. F. Hu. Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards. *Computer Communications*, 30(7):1655–1695, 2007.
6. R. Bouckaert. Bayesian network classifiers in weka for version 3-5-8, july 2008.
7. D. S. C. Ramos, J. C. Augusto. Ambient intelligence - the next step for artificial intelligence, *iee intelligent systems*, vol. 23, no. 2, march/april 2008.
8. F. G. Cozman. The interchange format for bayesian networks. <http://www.cs.cmu.edu/~fgcozman/research/interchangeformat/>.
9. R. Cucchiara, C. Grana, A. Prati, and R. Vezzani. Probabilistic posture classification for human-behavior analysis. *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, 35(1):42–54, Jan. 2005.
10. R. Cucchiara, A. Prati, and R. Vezzani. Posture classification in a multi-camera indoor environment. In *ICIP (1)*, pages 725–728, 2005.
11. Á. Fides-Valero, M. Freddi, F. Furfari, and M.-R. Tazari. The persona framework for supporting context-awareness in open distributed systems. In *Proceedings of European Conference on Ambient Intelligence (AmI08)*, volume 5355 of *LNCS*, pages 91–108, 2008.
12. M. Girolami, S. Lenzi, F. Furfari, and S. Chessa. SAIL: a Sensor Abstraction and Integration Layer for Context Aware Architectures. In *Proceedings of the 34th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2008)*, pages 374–381, Parma, Italy, September 2008. IEEE.
13. I. A. Group. Scenarios for ambient intelligence in 2010, european commission, 2001.
14. T. Gu, H. K. Pung, and D. Q. Zhang. A middleware for building context-aware mobile services. In *IEEE Vehicular Technology Conference*, pages 2656–2660, 2004.
15. X. He and P. Niyogi. Locality preserving projections. In *In Advances in Neural Information Processing Systems 16*. MIT Press, 2003.
16. D. Heckerman. A tutorial on learning with bayesian networks, , m. i. (ed.), learning in graphical models. kluwer, dordrecht, netherlands (1998).

17. S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen. The gator tech smart house: A programmable pervasive space. *IEEE Computer*, 1(3):64–74, 2005.
18. S. Kotsiantis. Supervised machine learning: A review of classification techniques, *informatica journal* vol. 31 (2007), pp. 249-268.
19. M. Madden. The performance of bayesian network classifiers constructed using different techniques, *proc. of european conference on machine learning, workshop on probabilistic graphical models for classification*, september 2003.
20. MESA Imaging. <http://www.mesa-imaging.ch/>, February 2009.
21. T. B. Moeslund, A. Hilton, and V. Krueger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2):90–126, November 2006.
22. S. Molla, M.M.; Ahamed. A survey of middleware for sensor network and challenges. In *Proceedings of International Conference on Parallel Processing Workshop (ICPP 2006)*, August 2006.
23. S. Pellegrini and L. Iocchi. Human posture tracking and classification through stereo vision and 3d model matching. *J. Image Video Process.*, 8(2):1–12, 2008.
24. M. Piccardi. Background subtraction techniques: a review. *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, 4:3099–3104 vol.4, Oct. 2004.
25. R. Poppe. Vision-based human motion analysis: An overview. *Comput. Vis. Image Underst.*, 108(1-2):4–18, 2007.
26. F. Porikli and O. Tuzel. Bayesian background modeling for foreground detection. In *VSSN '05: Proceedings of the third ACM international workshop on Video surveillance & sensor networks*, pages 55–58, New York, NY, USA, 2005. ACM.
27. H. Wang, S. Chen, Z. Hu, and W. Zheng. Locality-preserved maximum information projection. *Neural Networks, IEEE Transactions on*, 19(4):571–585, April 2008.
28. L. Wang and D. Suter. Analyzing human movements from silhouettes using manifold learning. *Advanced Video and Signal Based Surveillance, IEEE Conference on*, 0:7, 2006.
29. L. Wang and D. Suter. Learning and matching of dynamic shape manifolds for human action recognition. *Image Processing, IEEE Transactions on*, 16(6):1646–1661, 2007.
30. F. Wientapper, K. Ahrens, H. Wuest, and U. Bockholt. Linear-projection-based classification of human postures in time-of-flight data. *Systems, Man and Cybernetics, IEEE Transactions on*, Oct. 2009.