



DIPLOMA THESIS

**INTEGRATION OF COMPUTER VISION
IN SCENARIO RECOGNITION
BY MEANS OF SYMBOLIC PROCESSING**

Submitted at the Institute of Computer Technology,
Vienna University of Technology
in partial fulfilment of the requirements for the degree of
Master Science

under supervision of

O. UNIV.PROF. DR. DIETMAR DIETRICH
and
DIPL.-ING. WOLFGANG BURGSTALLER
Institute of Computer Technology
Vienna University of Technology

by

MANUEL PASCUAL GARCÍA-TUBÍO

Student ID 0627351

Vienna, November 2007

Abstract

This thesis represents a proposal for scenario recognition enhancement by means of a system that integrates computer vision. The work is based on the ARS-PC (Artificial Recognition System Perception) concept, which tries to build a representation of the real world with support of several sensors of different nature. This concept is brought into praxis in a real prototype that achieves thereby the recognition of predefined situation and scenarios. This thesis intended of providing the ARS-PC prototype with a computer vision system as an additional information source – among the other sensors previously installed – in order to enhance the mentioned recognition of scenarios.

Through this work the ARS-PC model is studied as a whole with the current configuration of the respective prototype. The design of the system is carried out considering on the one hand the necessary computer vision tools – in sense of image processing, objects recognizing, etc. – and on the other hand the different mechanisms needed to integrate this system in that one running at the ARS-PC prototype. From the implementation of this work results a system that takes advantage of the sensors installed in the prototype by combining them with the computer vision system. Furthermore, the installation described in this thesis provides practical insights about possibilities and limits of the resulting integrated system.

Abstracto

En esta tesis representa una propuesta para la mejora de reconocimiento de escenarios mediante la integración de un sistema de visión artificial. El trabajo está basado en el concepto ARS-PC (Artificial Recognition System Perception), que intenta crear una representación del mundo real con el soporte de multitud de sensores de diferente naturaleza. Este concepto ha sido llevado a la práctica en un prototipo real que consigue de este modo el reconocimiento de situaciones y escenarios previamente definidos. Esta tesis representa la propuesta de proveer el prototipo ARS-PC con un sistema de visión artificial como fuente de información adicional – entre los otros sensores previamente instalados – para mejorar dicho reconocimiento de escenarios.

A lo largo de este trabajo el modelo ARS-PC es estudiado en conjunto con la actual configuración del prototipo correspondiente. El diseño del sistema está llevado a cabo considerando, por un lado las herramientas necesarias para la visión artificial – en el sentido de procesamiento de imagen, reconocimiento de objetos, etc. – y por otro lado los mecanismos necesarios para integrar este sistema en el que está funcionando en el prototipo ARS-PC. De la implementación de este trabajo resulta un sistema que saca beneficio de los sensores previamente instalados combinándolos con el sistema de visión artificial. Además, la instalación descrita en esta tesis proporciona enfoques prácticos sobre las posibilidades y los límites del sistema integrado resultante.

Acknowledgments

I am indebted to professor Dietmar Dietrich, head of the Institute of Computer of Technology, who gave me the opportunity to contribute to the ARS project and motivated me to start my diploma thesis. I express special thanks to my supervisor Wolfgang Burgstaller who supported me especially in the last, most important phase of this work. In addition I want to extend my gratitude to the ARS project team, especially to Edgar Holleis and Andreas Richtsfeld for the pleasant teamwork and his patience during the whole thesis.

Thanks to my family, for giving me unconditional support and motivation since ever.

Index

Chapter 1	Introduction	1
1.1	A new Approach	2
1.2	Purpose and Problem Statement	3
Chapter 2	Framework	5
2.1	Artificial Recognition System	5
2.1.1	The ARS Model	6
2.1.2	Symbol Understanding	6
2.1.3	Symbol Level	7
2.1.4	Scenarios	9
2.1.5	Representation Symbols	11
2.1.6	Snapshot Symbols	12
2.1.7	Microsymbols	14
2.2	Applications Environment – Smart Kitchen.	16
2.2.1	Layout	16
2.2.2	Sensory Equipment	18
2.2.3	Octobus and database	20
2.2.4	SymbolNet	21
2.3	Computer Vision	23
2.3.1	Definition	23
2.3.2	Software	24
2.3.3	Hardware	25
Chapter 3	System Design	27
3.1	Applications	27
3.1.1	Person Tracking	27
3.1.2	Scenario Recognition	28
3.1.3	Child safety	29
3.2	Running Modes	29
3.3	Image Processing	30
3.3.1	Person Detection	31

3.3.2	Shape Distinction	32
3.3.3	Background Subtraction	33
3.3.4	Camera Calibration and SPD formula	34
3.3.5	Image Aberrations	35
3.3.6	Image Treating	36
3.4	Interconnectors	39
3.4.1	Two-way connection	39
3.4.2	Symbolic Processing	40
Chapter 4	System Integration	41
4.1	Interconnectors (iCon)	42
4.1.1	Sender	43
4.1.2	Receiver	43
4.1.3	NewSymbolMessage modules	46
4.1.4	Module: Proof Symbols	48
4.1.5	Module: Proof Person	49
4.1.6	Module: List	50
4.1.7	Module: Get Values	50
4.2	Image Processing (imgPro)	52
4.2.1	Door Finder	53
4.2.2	Face Detect	55
4.2.3	Size Estimator	57
4.2.4	Camera Calibration	62
4.2.5	Magazine Controller	65
4.2.6	Cup Detection	68
4.3	Interface	70
4.3.1	Camera Loop	70
4.3.2	Function Management	71
4.3.3	Live Mode	72
4.3.4	Database Mode	73
4.3.5	Recording Mode	75
4.4	Applications	77
4.4.1	Adult/Child Distinction	77
4.4.2	Scenario: Adult/Child makes coffee	79
4.4.3	Scenario: Person takes magazine	80
4.4.4	Scenario: Child near hot stove	82
4.5	Integration in ARS-PC system	82
4.5.1	ARS-PC structure	83
4.5.2	Integration procedure	85
Chapter 5	Results and Further Work	86
5.1	Results	86
5.2	Further Work	91

Chapter 1 Introduction

Video surveillance – surveillance from French “watching over” – has advanced further and faster in the period from 2001 to 2005 than in any prior comparable time period [Kru06]. Long ago, around the sixties, video surveillance began with simple closed circuit television monitoring, intended in the most of the cases for law enforcement, but rather used in exceptional situations, commonly by the police. Years later video cassette recorders hit the market, what meant surveillance could be preserved on tape as evidence. Slowly was video surveillance being integrated in different fields like traffic control or security systems in banks or public areas. However video surveillance made complete sense with the technologies advance during the computer revolution. The digital migration, the emergence of Internet and the prices dropping of the electronic devices derived in the integration of video surveillance in many other disciplines. IP cameras, servers, LANs, WANs, algorithms, etc. are all converging along technologies such as access control, life safety, intrusion alarms, etc., with the intent of configure fully integrated systems [Kru06].

On the other hand, and in a parallel way to the video surveillance, another discipline called building automation has been undergoing deep changes in the last decades. The term “automatic” derives from Greek *automatos* which is composed of the roots words *autos*- “self” and *matos*- “thinking, animated” [OED01]. This term is rather understood in sense of *mechanization*, or providing humans with machinery in order to replace human power by mechanical power [Goe06]. Thus building automation is a discipline that is intended to provide buildings with a certain grade of autonomy. Modern automated buildings combines the security system *with* the fire, heating, ventilation, air-conditioning (HVAC), lighting, and overall personal communication functions. Combining fire system with access control has the advantage of being able to automatically deactivate the alarm on doors when fire alarms sounds. The integration of the fire detection with HVAC can help control smoke by using advanced systems to contain it to a particular floor.

Video surveillance and building automation have one factor in common. They tend to be human-centered; the nature of these two disciplines has derived in an inevitable tendency of fusion between them. The integration of video surveillance in building automation provides the autonomous system with a grater support and vice versa – e.g. integrating a video surveillance system with the fire systems allows end users to view footage and discover the

cause of a blaze as well as monitoring and view the fire as it occurring [Kru06]. Building automation should achieve an efficient management of the resources and a support in the use of the technical systems. It should help in the energy saving as well in extending the lifetime of technical installations and at the same time provide comfort and safety for the people living in there.

1.1 A new Approach

Long gone are the days when video surveillance meant low-resolution, black-and-white or analog closed-circuit television. In recent years has increased the performance at the system level, faster microprocessors, larger memories and even wider buses. Due to these technological advances, the arising of the computer vision was achieved. Computer vision is a discipline that focuses on providing computers with the functions typical of human vision. By means of this technique a system can automatically detect scenes with people and vehicles or other targets of interest, classify them in categories such as people, cars, bicycles, or buses, extract their trajectories, recognize faces and arm positions, and provide some form of behaviour analysis. Such an analysis relies on a list of previously specified behaviours or on statistical observations such as frequent-versus-infrequent behaviours. The basic goal is not to completely replace security personnel but to assist them in supervising wider areas and focusing their attention on events of interest. Although the critical issue of privacy must be addressed before society widely adopts these video surveillance systems, the recent need for increased security has made them more likely to win general acceptance [AIP07]. Computer vision results in this sense another proposal of mechanisation, it provides great support in fields like industrial automation, robotics and biomedicine among others and of course it provides great support in the video surveillance. Once again the convergence of the technologies arises.

Building automation is a discipline that needs from many others to be affordable, including the concept of video surveillance and the implementation of computer vision. However it is not doubt that an enhancement of these systems is achieved by means of a greater knowledge of the surrounding world, not only in sense of designing those systems, but also in providing them with a greater perception of the environment. Derived from this last premise, results the question about how to provide those systems with a greater perception. Cheap sensors do not often accomplish the necessary tolerances or the respective accuracy. However, by means of the combination of several simple sensors – based on different technologies and physical effects – is possible to extract high-quality data. With the combination of the different sensors, the information provided does not involve dependence on the accuracy and neither on the function of these sensors anymore. Moreover although sensors breakdown, is possible to obtain useful data through the redundancy of the information obtained as well as improve the failure-tolerance [Ric07, Pra06]. Considering the use of several different and/or redundant sensors, is important to take into account that the always more increasing number of data-

points derives in a greater endeavour in order to obtain and process the information generated. Actually the challenge that faces building automation in the current days is the way to process the large amount of data, resulting from the always more used sensors. It is in this question where the ARS arises with a new proposal in order to enhance building automation systems. As conclusion of several researches, the ARS found an existing system which is capable to process large amounts of information and manage it by extracting important facts from those which are irrelevant; the system found is the human brain. Thus the ARS extrapolates models of perception from neuroscience, psychology and psychoanalysis into building automation in order to provide building automation systems with human perception capabilities.

1.2 Purpose and Problem Statement

Nowadays at the ICT (Institute of Computer Technology) stands a real prototype called the “Smart Kitchen” (SmaKi) which represents the projection of the ARS model in the real world, an experimental model of building automation or concretely in this case, domotics – home automation. This prototype is currently equipped with different sensors, which are managed in order to recognize situations and scenarios by means of a system which composes this ARS model. The system which manages the SmaKi achieves with accuracy the detection of some predefined scenarios. However this prototype is currently equipped with less than 100 sensors, which most of them are used for the only application of detecting the position of persons in the room. This situation already shows some benefits of using the ARS model in building automation, nevertheless this model is intended for much more; in fact it considers the further installation of a wider variety of sensors that would provide the system which a greater perception capability. This consideration represents the starting point for this thesis; the currently system lacks of a vision system. Taking this fact into account and considering the great support the video surveillance provides to the building automation, the purpose of this work derives into providing the SmaKi with a computer vision system. This system should be properly integrated in order to enhance the recognition of situations and scenarios and consequently demonstrate the profits of using the ARS concept in building automation.

Chapter 2 describes the “state of the art” or framework in which this work is placed. This chapter answers to the questions “how”, “where” and “with what” this system is developed. In the first part of Chapter 2 is introduced the *Artificial Recognition System* concept, which represents the basis of this work. It describes a new proposal for the processing of sensor-data and the recognition of situations and scenarios. The ARS model gives specific guidelines in *how* to develop the different functions, not only the way to process the sensory information, but also the mechanism to recognize the scenarios. The second part of Chapter 2 treats the applications environment, the Smart Kitchen, environment in which this work is focused, *where* the integration of this system will be carried out. The last part of this chapter

introduces the computer vision as a new data source in the SmaKi and describes the software and the hardware used to fulfil this task.

In Chapter 3 is made a study about in which applications the integration of the computer vision can contribute to the ARS. Due to the raising of the system the main application is the scenario recognition, in addition the system is focused in “person tracking”, and in “child safety”. Subsequently are described the different tools or procedures needed to provide the system with basic capabilities of artificial vision in order to accomplish the applications. The Chapter 3 concludes with the necessary components that have to compose the system in order to achieve a proper integration of the system at hand in the already – at the SmaKi – running system. The system design derives in the consequently system integration, which is the content of Chapter 4 . This chapter represents the proposal of a possible system development which achieves the purposes for which this work is conceived. The results of this system, the advantages and the barriers found through its development are finally shown in Chapter 5 .

Chapter 2 Framework

The next section 2.1 gives a superficial idea about the ARS model; concrete guidelines specify the right procedure to follow in system design. Although the idea is to build a modular and portable system, the environment (section 2.2) in which this work is focused, establishes a specific starting point which makes conditional the development of the work. Section 2.3 gives a brief introduction about Computer Vision and describes the different software and hardware used for carrying out this task.

2.1 Artificial Recognition System

In order to adapt building automation to the new and always more challenging society demands, and taking into account the vertiginous development of the technologies, in year 2003 a new research group was founded at the ICT by Prof. Dietrich, called ARS the “Artificial Recognition System” [ICT07, ARS07, Rös06]. In the endeavour to enhance building automation the ARS applies the newest theories of neuroscience, psychology and psychoanalysis described in [Die04] and [Bra04], which are intended to extrapolate methods of human perception and situation recognition into automation control. Taking this situation as basis, the first step given by the ARS was by means of the work carried out in [Pra06], which creates an abstract representation of the world in order to improve the perceptive capabilities of automation systems [Rös06].

The next section 2.1.1 introduces the two groups of research in which the ARS was split, ARS-PsychoAnalysis (ARS-PA) and ARS-PerCeption (ARS-PC) and gives a bit deeper description about the system described in [Pra06]. In section 2.1.2 is described the symbol concept, since is an important component involved in [Pra06] often used throughout that work and consequently throughout this thesis. From this symbol concept results the necessity of creating a hierarchical structure in which different kind symbols are placed according to specific criteria, which is mainly the content of section 2.1.3. Finally in conclude this chapter is given a description about some of the already defined symbols and the existing connection among them.

2.1.1 The ARS Model

Consequence of the “Smart Kitchen” project explained in section 2.2 was the branch of the ARS in two different – however closely linked – research groups, the ARS-PA and the ARS-PC. The ARS-PA proposes different control architectures based on concepts of psychoanalysis and other areas of cognitive science, in order to give greater autonomy to control systems in sense of decision making [Rös06]. Nevertheless to build a consistent system, the ARS needs from another discipline to extract the necessary information of the real world for the further decision making. The ARS-PC, based in neuropsychology, faces this necessity by designing a system based in human like perception, which by means of the implementation of several sensors, extracts the essential characteristics of a situation [Pra06].

Nowadays building automation is supported by the great advance of the technologies, current systems are composed by thousands of different sensors and actuators to interface with the real world, the possibilities of designing multiple data source sensor networks are manifold. Using more sensors the capability of perception of the environment increases. Nevertheless, the more information available, the greater necessity to distinguish between important and unimportant information [Pra06]. Pratl, after several studies, concludes that the most suitable system to carry out such a task is the human brain, since it is a “system” that process constantly incoming data and analyzes the important information in order to build a representation of the world [Pra06]. Applying this idea, the work developed by [Pra06] represents the starting point for the design and the implementation of the system.

The ARS-PC group, impelled by the researches in [Pra06], has given rise to the implementation of a real prototype at the ICT, the assembly of the Smart Kitchen, in which the ideas of Pratl are brought to the praxis [Goe06]. Thus [Ric07] continued this project in his diploma thesis, resulting in the current ARS-PC system which manages the operation of the SmaKi and represents the real basis for the integration of the system developed through this work, the ARS-PC Vision. Throughout this thesis, these two systems, the ARS-PC system and the ARS-PC Vision, will be clearly differentiated. Although the purpose of this work is to assemble only one system that controls the SmaKi, is important to differentiate them in order to explain the proper integration of the system developed in here.

2.1.2 Symbol Understanding

Before going deeper in matter is important to describe what the term “symbol” means, since it represent in the context of this thesis much more than the common meaning of the word. Etymologically the term symbol (*symbolon*) stems from the late Greek, which leads to the word *symballain* what literally means “to throw together”. As the Encyclopaedia Britannica says, a symbol is “*an arbitrary or conventional sign used in writing or printing relating to particular field to represent operations, quantities, elements, or qualities*” [BOE07]. This last

definition could approach to the meaning of the term symbol used in this work, however the real concept of this word is given with the raising of the ARS model given in [Pra06].

Pratl introduces the term symbol after considering the necessity of resorting to the *sensor fusion*, in order to distinguish the sensor layer from the applications. Sensor fusion is defined in [Elm02] as:

[...] the combination of sensory data or data derived from sensory data in order to produce enhanced data in form of an internal representation of the process environment.

That means that, for a greater perception of the environment – i.e. an accurately system's representation of the real world – applications should use more than one sensor to gain redundant information from the physical world. Each piece of information originated from the sensors is what in [Pra06] is defined as *symbol*. The symbolic representation of events, objects and information in general, is what in the ARS systems is called as *symbolization* [Goe06].

2.1.3 Symbol Level

Due to reasons described in [Pra06], the defined symbols are arranged following a hierarchical architecture, different levels that represent not only more grade of complexity, but also different states of perception. In Figure 1 are depicted the symbols (represented as cuboids) and their relation among the levels, the higher the level, the greater the volume and sophistication. The system comprises three¹ different levels, the *microsymbol* level, the *snapshot symbol* level and the *representation symbol* level.

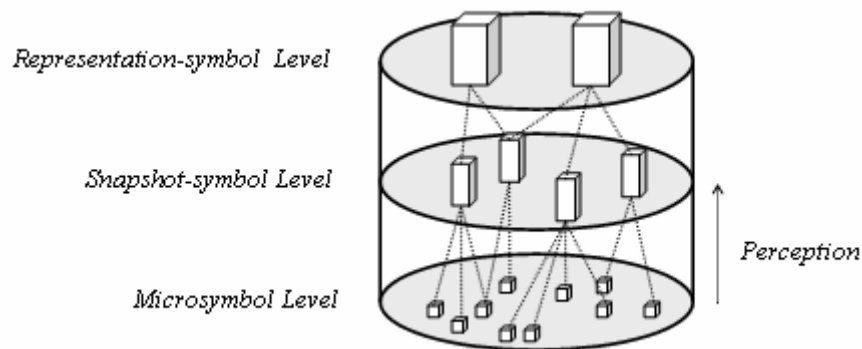


Figure 1: Symbol levels and correspondences in the ARS model [Pra06].

1. Actually the system described in [Pra06] is composed by more layers than those explained in this work, but these are those necessary and enough for the design and development of this system.

This distribution makes analogy to the human process of cognition, in order to apply it to building automation (see [Die04, Bra04]). The first layer is the one directly connected with the sensors, therefore the large number of cuboids and the less complexity of these symbols. On the other hand is the top level, which represents the output of the system, how the system interprets the surrounding world, the world representation.

Microsymbol Level – The symbols in this level are directly generated through sensor values, it is in this level where the symbol data-flow starts. Due to the little information processed in these symbols, normally binary, they are designated as *microsymbols*. When the environment changes and with it a sensor value, microsymbols are generated, deleted or modified. For example if a person goes through the room, several sensors are driven and thereby several microsymbols are fast generated and deleted. Since this level is directly connected with the sensors, if the number of sensors rises, the number of microsymbols available rises too (sample of this connection is shown further in Figure 20 in section 2.1.7). Thereby is possible to generate redundant information, since different sensors can provide same information. The movement of a person is detectable with cameras, tactile floor sensors and/or movement detectors and each of them would provide support in the generation of the corresponding microsymbol.

Snapshot Symbol Level – Symbols in this level are based on one or more microsymbols and compose the previous layer for the *Representation Level*. The symbols in this level are designated *snapshot* since they represent a part of the environment in a specific moment. All the snapshot-symbols together represent the real world understood in a concrete moment.

Representation Symbol Level – This level stands between the Snapshot Level and the *Scenario Level*. These symbols represent the surrounding world with the information provided in the layer underneath. Unlike the previous two layers, the representation symbols have a long lifetime and are rarely generated or eliminated. Representation-symbols contain all the information about the state of the real world and build this way a consistent and continuous representation of the environment. While the representation symbol exists its properties are regularly updated, they represent the environment in a concrete moment but they also contain information of the past.

Scenario Symbol Level – This level is placed at the top of the pyramid, where *scenario – symbols* are generated. They describe a concrete situation or a series of situations in the real world. These symbols represent the real aim of the whole system, since they compose the different purpose applications.

The current system is composed by a 30-symbols alphabet [Ric07]. In the next sections are explained only the symbols that are necessary to the realization of this work. Starting from

those which contain more information, the scenario – symbols, until reaching up to those which are the base of the system, the microsymbols.

Figure 2 describes the diagram used to explain the different symbols in the next sections; it indicates the name of the symbol and the properties that characterize it, as well as the level (“lvl”) in which the symbol is produced – “sc” stands for scenario-symbol level, “rp” for representation-symbol level, “ss” for snapshot-symbol level and “ms” represents the microsymbol-level.

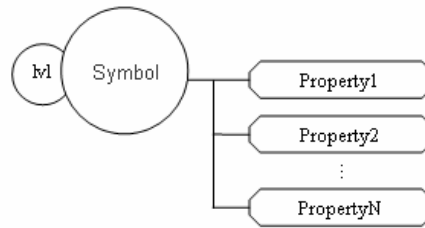


Figure 2: Level – symbol – properties diagram.

The symbols described next attempt to build the representation of the environment. These symbols and also their properties are fix defined for this system, but are able to be modified in further works. They are defined taking into account the sensors available, if new sensors are installed in the system, the number of symbols can increase and their properties modified or expanded.

2.1.4 Scenarios

Meeting – The scenario meeting describes one get-together of persons at one table. That means that at least two persons must take part in this scenario. The scenario does not consider what the persons are doing during the meeting. If these persons meet near a coffee machine or bookshelf, the scenario will be generated too. As explained in Figure 3 this is a scenario – symbol and contains, as only property, the place where the meeting occurs.

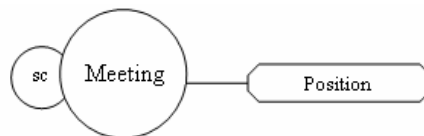


Figure 3: Scenario – symbol "Meeting".

Person and object – The scenario “Person and object” describes the manipulation of one object by one person. The object represents any item detected by the system. The scenario should be generated when a person brings an object into the room, as well as when the person takes it from the room away. Represented in Figure 4 is the only property that belongs to this symbol, which is the position where the object was left.

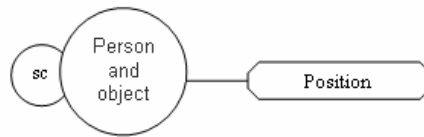


Figure 4: Scenario – symbol “Person and object”.

Person makes coffee – This scenario will be generated when one person makes coffee. As represented in Figure 5, the name of the person, where and when the coffee was made, are properties identified as part of the scenario. The scenario will exist for that person until he/she leaves the room.

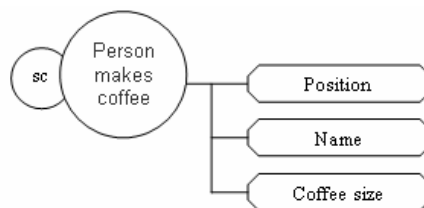


Figure 5: Scenario – symbol “Person makes coffee”.

Child near hot stove – When a child is near a hot stove, it can represent a dangerous situation for the child. This is the case when no adult is in the room. This scenario should describe a situation of danger. Figure 6 describes the property that characterizes this symbol; the position where the event occurs.

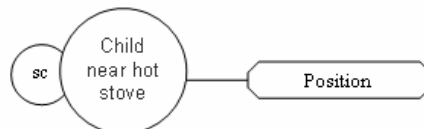


Figure 6: Scenario – symbol “Child near hot stove”.

Child makes coffee – As considered in the previous situation, permitting a child to manipulate the coffee machine represents a dangerous situation for him. The scenario “Child makes coffee” represents an unwanted situation for the child. The property contained in this symbol is the position; represented in Figure 7.

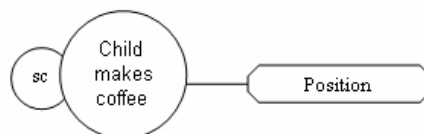


Figure 7: Scenario – symbol “Child makes coffee”.

Person takes magazine – The scenario “Person takes magazine” describes the situation in which a detected person modifies the “state” of the bookshelf. This scenario should be generated when a person takes a magazine away from the bookshelf, as well as when the person puts it back in the bookshelf. Represented in Figure 8 is the only property that belongs to this symbol, which is the position where the magazine was taken, actually the position of the bookshelf.

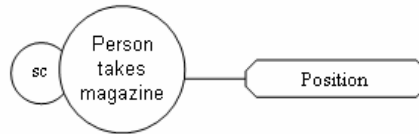


Figure 8: Scenario – symbol “Person takes magazine”

All the scenarios mentioned above are part of the already defined symbols in the ARS-PC; as exception of this last one scenario-symbol “Person takes magazine”. This last symbol represents a new adding to the actual system thanks to the integration of the computer vision system developed in the work at hand.

2.1.5 Representation Symbols

Person – Since persons are the main target of the system, the one which takes part in all applications, all the symbols in the adjacent levels are linked with the symbol Person. As described in Figure 9, this symbol is placed in the representation layer and their properties are the called, position, usage, near and name. The three last properties are optional. The position describes the actual location of the person in the room, the usage property is generated if the system detects subject manipulating some object, and the property near appears when the person is close to a specific piece of furniture or object from the room. And finally, considering that the person can be detected and recognized there is a possible property name. Until this symbol exists, those properties can be updated considering what the person does in the room.

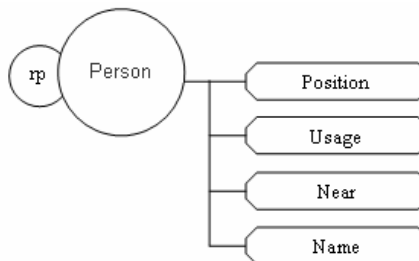


Figure 9: Representation – symbol Person.

Child – The symbol Child as the symbol Person, is the basis for one of the main applications of the system; in this case, “child safety”. This symbol is used to distinguish adults from children and is the one needed to compose the subsequent scenario – symbols: “ChildNearStove” and “ChildMakesCoffee”. The symbol Child belongs to the representation level and contains the property near, what is represented in Figure 10.

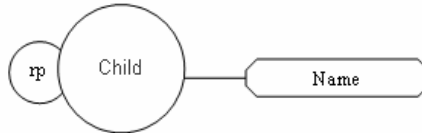


Figure 10: Representation – symbol Child.

Hot stove – This symbol is generated when the stove becomes hot. This symbol, combined with the previous one generates the “Child near stove” symbol. As described in Figure 11, it is one of the representation – symbols and its property position determines where the stove is situated in the room.

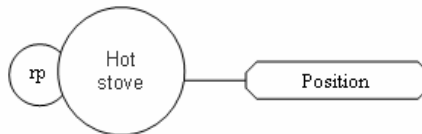


Figure 11: Representation – symbol “Hot stove”.

Make coffee – The symbol “Make coffee” represents the preparation of coffee. This symbol contributes to the generation of the symbols “Person makes coffee” and “Child makes coffee”. Represented in Figure 12 is the representation – symbol and its property position.

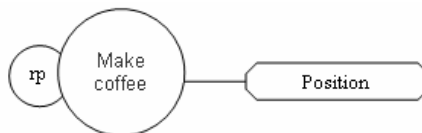


Figure 12: Representation – symbol “Make coffee”.

2.1.6 Snapshot Symbols

Gait – When a person walks through the SmaKi, activates a succession of sensors. The snapshot – symbol Gait represents a succession of known sensor values, which allows to follow the route of the person along the room. Figure 13 drafts the five properties that can appear together with the symbol Gait. The property “start position” contains where the gait starts and the property position is the actual location of the person. Velocity, direction and acceleration can appear within the symbol in matter too.

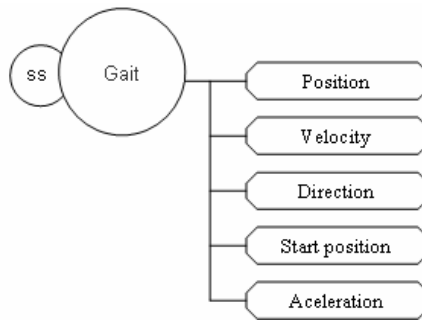


Figure 13: Snapshot – symbol Gait.

Footstep – This symbol represents a footstep from a person. When a person walks over the floor sensors he or she leaves a specific pattern as he/she presses and leaves the sensor. This pattern must be recognized and represented through this symbol. This symbol will help to distinguish persons from other objects in the room. Located in the snapshot layer this symbol has always the property position. The property velocity and direction can appear too, as shown in Figure 14.

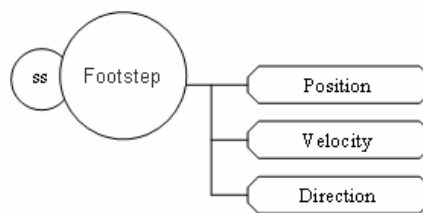


Figure 14: Snapshot – symbol Footstep.

Item – The symbol Item represents a concrete fixed object in the room that is considered as potential cause of scenarios. This symbol is connected to the symbol person and provides it with worthy information about its activities. Represented in Figure 15 is the symbol item, which belongs to the scenario level and contains the property Position, which indicates the coordinates of this object in the room. It also contains the property Item which provides the type of item which is being treated.

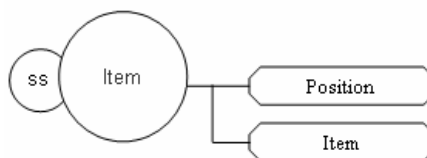


Figure 15: Snapshot – symbol Item

All symbols of type Item have no functionality that can be detected by the system. This is valid for example for the bookshelf, the copy machine and the table. Items which have

functions that are considered by the system are represented with other symbols. A sample of this fact are the symbol “Coffee machine”, Fridge or the symbol Door, represented in Figure 20 and detailed described in [Ric07].

2.1.7 Microsymbols

Object – Generated as microsymbol Object is the one produced by the system when it detects anything in the room and cannot recognize what it is. The way the objects are detected is by mean of the tactile floor sensors, hence the imprecision of this information; any object, as well as a table leg or in a first moment a person, are detected this way as an object and the microsymbol is consequently generated. As described in Figure 16, the property belonging to this symbol is the position where the object is detected.

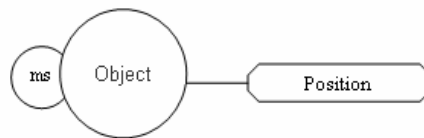


Figure 16: Microsymbol Object.

Movement – The symbol is generated when movement in room is detected. It is based in the binary sensor values provided by the motion detectors that control concrete regions of the room. The only property that is contained in this microsymbol is the property position; shown in Figure 17.

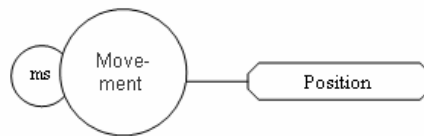


Figure 17: Microsymbol Movement.

Door status – The symbol “Door status” represents the state of the door at all times, it is a symbol which is always present. Since the door is a fixed element in the room its property position, shown in Figure 18, is invariable too. At the moment the program starts, this symbol is generated with all its properties.

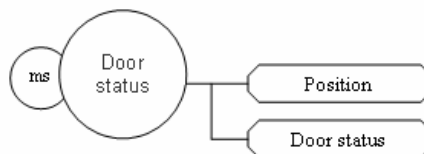


Figure 18: Microsymbol “Door status”.

Identify – The symbol Identify should facilitate the recognition of predefined Persons through concrete person features. This symbol is used for the generation of the symbol “Known person”. This microsymbol has the properties Position and Name, what is depicted in Figure 19.

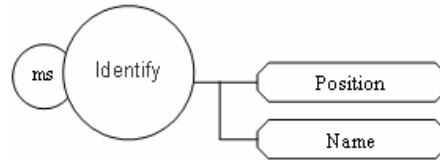


Figure 19: Microsymbol Identify.

The complete symbol tree with the respective connections among the different levels is shown in Figure 20 which represents the *ARS-PC Graphical Representation* for the SmaKi. This is part of the graphical layer of the already implemented ARS-PC system developed in [Ric07].

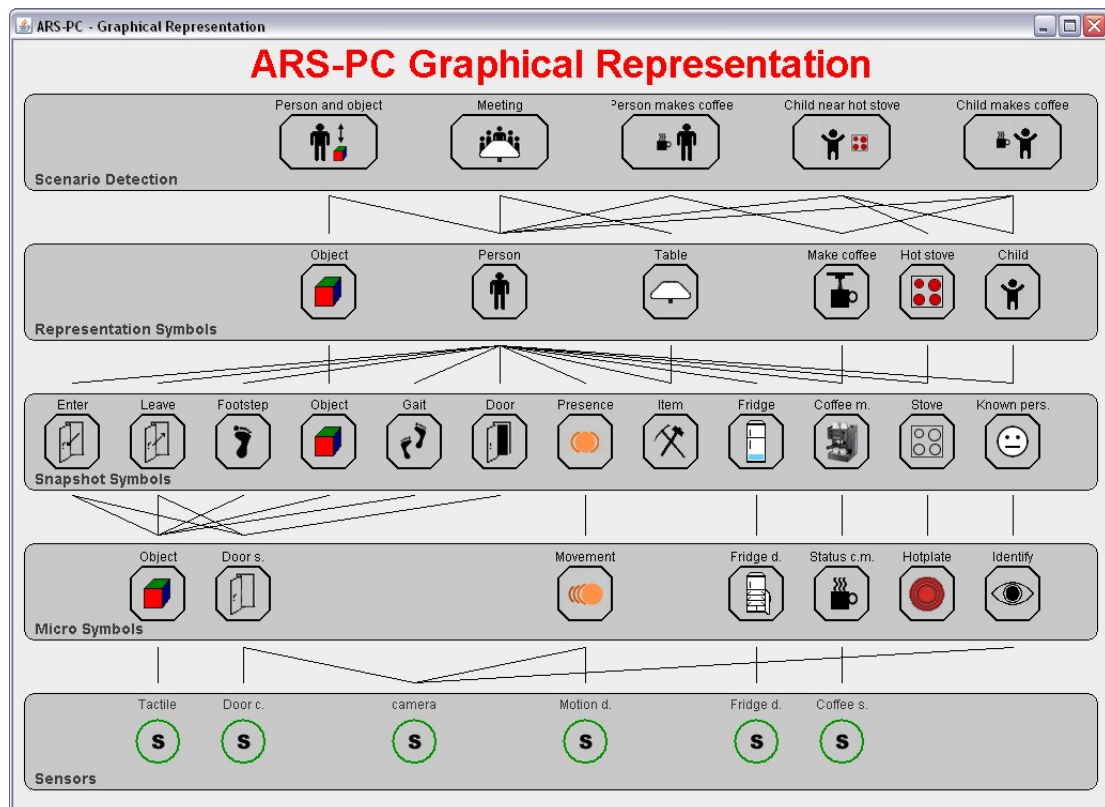


Figure 20: ARS-PC Graphical Representation. Symbol interconnections.

The graph depicted in Figure 20 makes analogy to the symbol level model described in section 2.1.3 (see Figure 1). The graph at hand is composed of five levels. The one at the bottom represents the sensor layer, in there are represented all the available sensors at the SmaKi,

tactile floor sensors, door contact switch, cameras, etc. The next level, the one connected with the sensor layer is the microsymbol level. In this level are represented some of the mentioned symbols, like the microsymbol Object or the “Door Status” symbol, as well as others like the microsymbol Movement or the symbol “Coffee machine status”, each and everyone of them defined directly by the respective sensors. The level above is the snapshot-symbol level, in this level there are symbols like the explained snapshot-symbol Gait or others like the snapshot-symbol Footstep – this represents the footstep from a person, recognized since the person leaves a specific pattern as he walks over the floor sensors (the exact definition of each and every symbol is described in [Ric07]). The next level depicted is the representation-symbol level, here is possible to observe the mentioned relevance the symbol Person means in the system, all the symbols from the snapshot-level are connected to the symbol in matter, as well as this symbol is involved in the generation of all the symbols in the next level. The next and highest represented level is the scenario-symbol level where are represented all the mentioned symbol-scenarios in section 2.1.4 (as exception for the scenario “Person takes magazine” that what was not implemented as [Ric07] developed this application).

2.2 Applications Environment – Smart Kitchen.

The Smart Kitchen started as project to prove the profits of using many small-inexpensive devices integrated with a fieldbus network in domotics [SRT01] . With this idea, the project was continued with the design and assembly of a prototype [Goe06], basing it on the principles of the ARS system described in [Pra06]. Nowadays the SmaKi is a real prototype which, equipped with several sensors, represents the test-environment for the ARS-PC.

In this section the SmaKi is described with its respective components, starting with the layout of the room in section 2.2.1, then in section 2.2.2 the sensor equipment and distribution of the SmaKi, and finally the different hardware installed (section 2.2.3) and software used (section 2.2.4) to make the system work.

2.2.1 Layout

After the assembly of the SmaKi, the result layout of the prototype ended as shown in Figure 21. Following this figure the corresponding elements of the room are described next clockwise: On the left side of the room there is a door (A), the only one in the SmaKi. Next to the door, on its left side, there is a sink (B) and a dishwasher (C) underneath. Beside the sink there are a stove (D), a coffee machine (E), and a fridge (F). In the piece of furniture next to the fridge (G) stands the hardware for the processing of the data coming from the different sensors. Keeping the same direction, now in front of the door, there is table (H) and close to it (I) the only window in the room. There are, against the right wall, a bookshelf (J) and a copier (K). And finally, in the corner, there are some shelves.

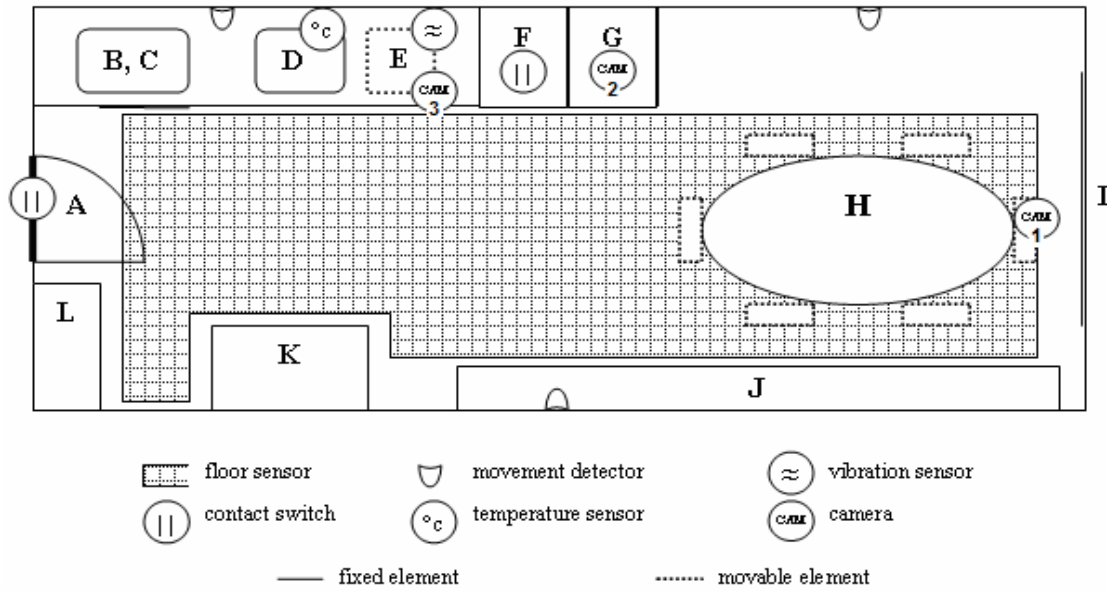


Figure 21: Graph in plan view of the SmaKi with sensors.

The different sensors shown in Figure 21 are specially placed to control specific elements or regions in the room that are full of interest for the applications this environment is focused. The ARS indicates the advantages of combining the information of the different sensors. In order to assemble new sensors in the SmaKi is important to consider which regions are already covered and which not. For sensors like the tactile sensors or cameras the approximate metrics of the room are valuable too. Figure 22 shows some room dimensions of interest in the SmaKi, necessary for the ARS-PC Vision design.

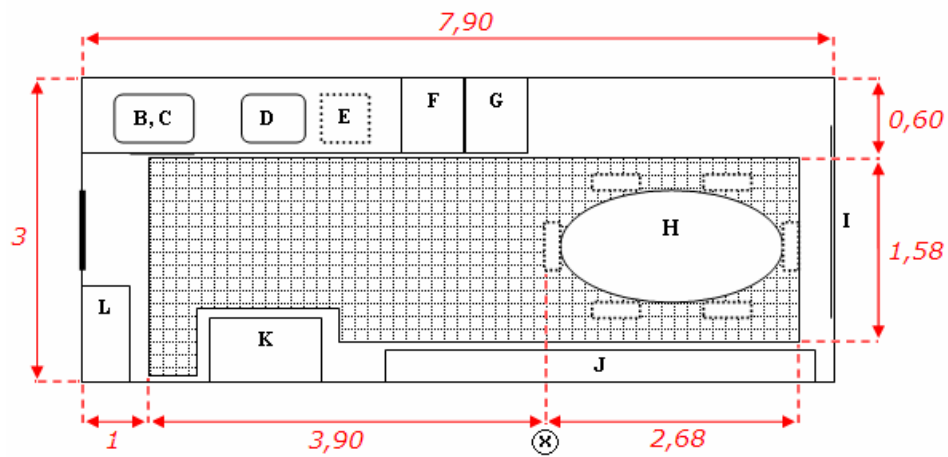


Figure 22: SmaKi plant view with metrics of interest

2.2.2 Sensory Equipment

As mentioned above, the SmaKi must be assembled with some small sensors which provide the information. These devices are the basis of the system, the transducer of the real-world events into manageable information. The different sensors and their situation in the SmaKi are represented in Figure 21.

Floor sensors – These sensors are placed almost all around the room, taking care in specific areas which ones can be of special interest; a total number of 94 sensors each with 600x175 mm dimension. Actually the (approximate) distribution of the tactile floor sensors is represented in Figure 21.

Floor sensors work when a pressure is exerted on them and provide a binary signal when this occurs. Their position is given in three values: POS1, POS2 and POS3, which is enough information to determine the coordinates of the sensor in the room; since, as shown in Figure 23, a square is directly described with these three points.

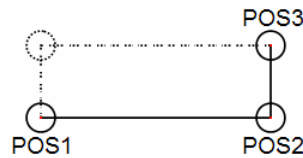


Figure 23: Floor sensor with three point position.

Switch contacts – Switch contacts are used to detect the state of a device; for example “on – off” or “open – closed”. As represented in Figure 21 there are two switch contacts in the SmaKi, one in the main door and the second one in the fridge. Both of them to determine if each door is open or closed.

Vibration sensor – Attached to the coffee machine there is a vibration sensor. When coffee is prepared the coffee machine vibrates, and when it finishes serving it vibrates again. With this information it is possible to determine, not only when the machine is working, but also to estimate how much coffee is served. The sensor only detects vibrations greater than 1 m/s^2 .

Motion detector – In the SmaKi there are three motion detectors from type PIR (Passive InfraRed). They are specially placed to cover the whole room, as described in Figure 24, one on the right side of the room and the remaining two on at the opposite wall. Since the detection region describes a pyramidal form, these three detectors are enough for an optimal detection coverage.

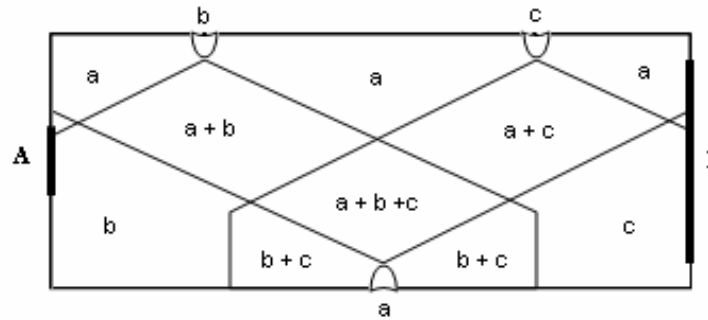


Figure 24: PIR coverage in SmaKi in plant view.

These PIR devices work with the infra-red spectrum and detect heat in movement sources. This is the reason why they are called *passive*, since they do not emit any radiation, only wait to detect for detecting some heat variation.

This kind of detectors is limited by its sensitivity, in two aspects chiefly. If a person moves very slowly, it means that the heat variation is very small and the device will not detect it. If the object to detect is very small, the amount of heat will be negligible for the PIR and the device will not react.

Temperature sensor – To control the state of the stove, whether it is hot or not, a temperature sensor must be installed there. Although at the moment there is no temperature sensor assembled, the system in which this work is based [Ric07], considers the future installation of this sensor in the SmaKi and was implemented taking care of this fact.

Cameras – All the sensors mentioned above provide very useful information, but only binary information. For more sophisticated detection, cameras are needed; for example, human activities or object shapes are only detected this way. Cameras provide a great amount of information and their application possibilities are incalculable. Thereby with the combination with the rest of the sensors the resulting information is much more robust than before.

Three USB WebCams are assembled in the SmaKi: CAM1, CAM2 and CAM3. They are placed specifically to control different regions of the room. As represented in Figure 25 pointed in blue, keeping the notation from Figure 21, CAM1 is placed focusing on the door (A) right on the other side; centred a) and fixed to the ceiling b). CAM2 is located on the top of the fridge (F) and is focusing to the bookshelf (J). The last camera, CAM3, is focused downwards, situated over the coffee machine (E). The specifications and a wider description of these cameras are given later in section 2.3.3.

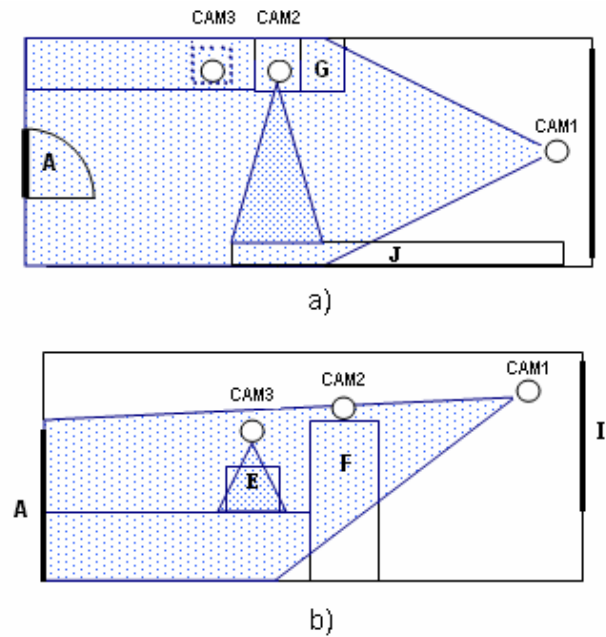


Figure 25: Views from cameras 1, 2 and 3. a) SmaKi plant view. b) SmaKi side view.

2.2.3 Octobus and database

Except for the cameras, all the sensors in the Smart Kitchen can be considered simple, in the sense that they provide only binary information. Due to this fact a specific hardware was necessary for the sensor data analysis. Hence, this hardware, called *Octobus*®, and its corresponding software, was developed at the ICT in collaboration with “haag.cc embedded systems & it consulting GMBH” as part of a previous diploma thesis [Goe06]. The Octobus is composed by two main modules, the CPU module and the Octobus module. The CPU module is an IBM processor equipped with an embedded Linux operative system. For the communication with the ARS-PC system it is equipped with two net interfaces and a serial port. The sensors are able to be connected through specific extension modules designed as *OctoIO*. These modules have 12 digital inputs and outputs and are read and written through a bus system. It is possible to connect up to 16 OctoIO modules, which provide the Octobus with a total number of 192 digital inputs and 192 digital outputs.

As shown in Figure 26, the Octobus is the interface between the sensor network and the ARS-PC System. If one sensor value changes, a message is sent from the Octobus to the ARS-PC System and those changes are written in the *Sensor-Database* at the same time. The Octobus needs 70 ms to check a sensor change and send the respective message. Although the ARS model does not consider any data storage for long periods, two databases were mounted in the SmaKi, one for the storage of sensor information and the other one for the storage of the generated symbols.

With the implementation of the Sensor-Database all the information coming from the sensors produced in each scenario can be stored. Since the information is stored in the database, the ARS-PC System can always reproduce those scenarios. This provides the possibility to test the ARS-PC and check the effects of the changes when programming. Due to the great amount of symbols the ARS-PC processes, an additional database is necessary in which all the symbols can be stored, the *Symbol-Database*. All the symbols with their properties can be stored there by the ARS-PC System. With this data the ARS-PC can generate, actualize and delete the symbols.

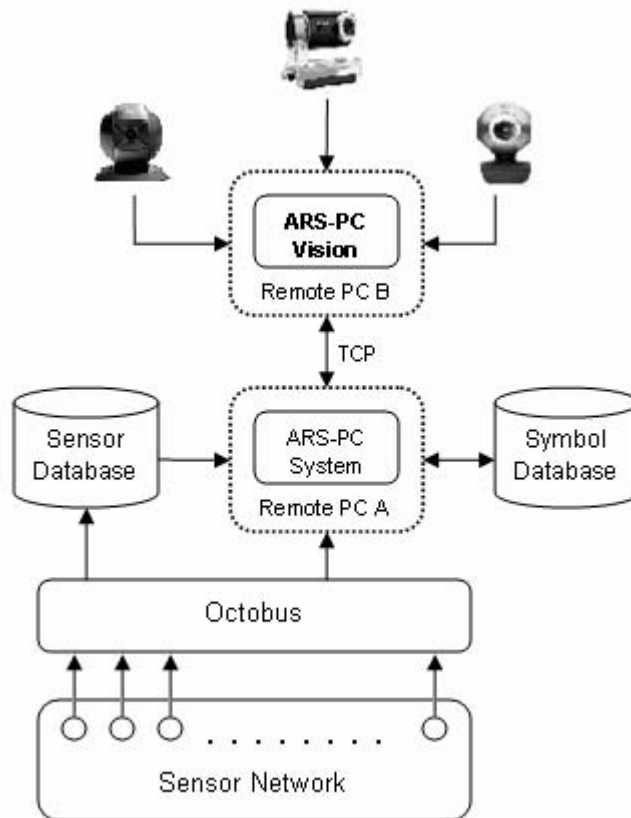


Figure 26: SmaKi hardware connections.

2.2.4 SymbolNet

The ARS framework uses a software-package called SymbolNet to process and transmit data. It was developed at the ICT for the ARS project in a work carried out by [Hol06]. This package contains several functions to process symbolic information. In this section are described the most important concepts and notations considering the development of this work. Among the different components composing the SymbolNet there are *Symbol-Beans*, with the possible fields and the different properties together with the *Messages*.

Symbols (Symbol-Beans) – In SymbolNet symbols are understood as data packages. Each package is composed of different fields. There are some fields that are fixed for all the symbols and others that can vary depending on the characteristics of each symbol. The possible fields appearing in a symbol are:

- *ID*: When a new symbol is generated it receives an unequivocal identification number. This way each symbol is differentiated from the rest.
- *Type*: The type describes which level the symbol belongs to: micro, snapshot, representation or scenario (see section 2.1.3).
- *Class*: Each sort of symbol in the SymbolNet must be able to be placed in an unequivocal symbol-class. The class field specifies the properties that are obligatory and which are optional in the symbol.
- *Timestamp*: It is the point of time in which the symbol was generated. If the symbol is updated the timestamp will change into the current point of time when the update occurs.
- *Lifetime*: If no lifetime is given, the symbol will exist until an *ExpireSymbol-Message* for this symbol is generated. If a lifetime is specified it will exist until its lifetime, given in milliseconds, expires.
- *Properties*: This field represents a list of symbol-properties. A symbol can have properties but it is not obligatory.

Symbol-Properties – Like the symbols, the symbol-properties are given with a specific structure with different fields. The different properties are:

- *Class*: Each property-symbol must be able to be located in an unequivocal property-class which gives the semantic meaning of the property.
- *Schema*: Each property-class permits different schemas for one property. In a property the schema gives the meaning of each value.
- *Confidence*: The confidence gives the reliability of a property-value.

Symbol-Messages – The communications between the different symbolic processing modules is carried out by means of these messages:

- *NewSymbol-Message (NSM)*: This message transports a new symbol with all the necessary data to another processing module.
- *UpdateProperty-Message (UPM)*: With this message a module can change the property of an existing symbol.
- *ExpireSymbol-Message (EM)*: If a module needs to delete a symbol this message will be sent to all the modules where the symbol exists, those which received the NewSymbol-Message for that symbol.

- *Heartbeat-Message*: These messages are used to synchronize times among processing modules. These messages are sent by a module which has access to a real-time clock.

SymbolNet uses TCP sockets to send and receive the messages, which provide the possibility to exchange information among different computers via Ethernet. This fact means a great advantage since the ARS-PC System can be installed in different computers and it is possible, this way, to share and optimize the computer performance. For transmission and reception SymbolNet encodes and decodes messages in a DER way [Dub00].

2.3 Computer Vision

The whole system explained above is a robust system, which, provided with the set of sensors, can identify accurately the different situations it was designed for. But although it was considered in previous ARS works [Pra06, Goe06, Ric07] up to now the system lacks a vision system. It is known that the human sight sense provides approximately the 50% information got by the different five senses [Big06], it is by providing the system with a computer-vision that it becomes more robust. Many of the applications already existing can be supported by the vision system and many others can be only developed this way.

This section tries to give a general idea about the concept of the Computer Vision and what is intended for. In sections 2.3.2 and 2.3.3 the different tools used to work in this field, software and hardware are described.

2.3.1 Definition

To find an appropriate definition for Computer Vision is a complicated task, since it is an immature science and covers many different fields. The first significant works related to Computer Vision started only 30 years ago [Big06], when computers started being capable to process great amount of data such as images. Fields like artificial intelligence, neurobiology or physics are strongly related to the computer vision; however the application in other complete different disciplines are manifold; proof of this fact are for example the system developed at the University of Western Ontario for *Tracking “Fuzzy” Storms in Doppler radar Images*, the *3D Velocity Fields from Flow Tomography Data* developed at the Faculty of Civil Engineering and Geosciences (University of Technology) in Netherlands or the *Analyzing Size Spectra of Oceanic Air Bubbles* project developed at the Heidelberg University, Germany [JH00]. Some definitions of the term Computer Vision, given by prestigious investigators in this field are shown next:

- Science which develops the theoretical and algorithmical bases to obtain information from the real world through one or several images [HS92].

- Discipline which develops systems with the capability to interpret the content of natural scenes [Cas96].
- The goal of computer vision research is to provide computers with humanlike perception capabilities so that they can sense the environment, understand the sensed data, take appropriate actions, and learn from this experience in order to enhance future performance [SCGH06].

Looking at these definitions it is proved the risk that involves giving an exact definition due to the wide coverage of this field.

2.3.2 Software

Working with computer vision requires many complex algorithms. Study them in depth and integrate them in a conventional program is a hard task and not the one of this work. To address *reusability* and *efficiency* is necessary to resort to standard data structures and implementations of classic algorithms, what is affordable with comprehensive code libraries [MK04]. Nowadays there are some libraries focused in computer vision, for different platforms and different programming languages. Samples of this fact are the already available LTI-Lib [LTI05] – object oriented library in C++, tested in Linux and Windows NT LTI05 – which stopped its development in year 2005, or VIGRA – which runs with some versions of GNU and Microsoft Visual C++ [VIG06] - that is neither in development anymore. For the Java programming language are some libraries available too, like the JAVAVIS [JAV07] or the VASE package [VAS06], however even these libraries were not more than merely attempts of a consistent end-software package. It is important to remark that all libraries mentioned here are referred to open source libraries, fundamental question to develop experimental setups like the one developed in this thesis. Among these variety of libraries there stands out the one developed by Intel, the OpenCV.

The OpenCV is a set of libraries appointed chiefly for image processing. Its designation refers to two concepts: *Open* is because it is an open source library. A main reason for working with it, because even though OpenCV is originally developed by Intel, at the moment are many updates, patches and samples developed by selflessly collaborations. The second part which is *CV* is, as expected, for computer vision [ORM01].

Since OpenCV started its development in 1999, is the computer vision package with the widest variety of tools for image interpretation existing nowadays, it contains over 500 functions; which is the second reason for having chosen it. These libraries are written all in C and C++ and are available in Windows, Linux and MacOXD [OCL07]. Is compatible with Intel Image Processing Libraries (IPL) and composes basic operations like binarization, filtering, image statistics or pyramids, although it is manly focused on techniques like optical flow tracking, shape analysis, object segmentation or image 3D reconstruction. The algorithms are based in flexible data structures coupled with IPL structures; more than half of

the functions have been optimized taking advantage of the Intel's architecture [ORM01]. OpenCV is provided with a graphic interface called HighGUI, but its only intended for quick software prototypes and experimentation setups development.

2.3.3 Hardware

There is an innumerable variety of products available to work with computer vision. Such frame grabbers, DSPs, high-resolution CCD cameras, etc. the possibility of creating a great sophisticated video system is there with the corresponding investment of money. However, as mentioned in section 2.2 and remaining faithful to the SmaKi project principles, the idea is to use small and inexpensive devices. Hence the ICT provides three USB WebCams for the realisation of this work and the consequent assembly of the cameras in the kitchen. USB cameras are furthermore more suitable for experimental setups due to its easy configuration and its intrinsic portability, always better integrated in the current operative systems. These three cameras are depicted in Figure 27 and the following paragraphs provide a brief description.

Creative Live! Cam Optia AF – This camera (depicted in Figure 27 c) is the most sophisticated of the three. It is mounted with a 2 megapixel sensor and high quality precision lens. The highest still image resolution is 8 megapixels (3200x2400) enhanced with software and the video resolution is 1600x1200. The camera has an auto-focus device and a High-Speed USB 2.0 (backward compatible with USB 1.1) interface [CWF07].

Logitech QuickCam® Communicate™ STX – This camera is equipped with a high quality VGA sensor with RightLight™ Technology. It mounts fixed focus, has a video capture resolution up to 640 x 480 pixels and a maximum rate of 30 frames per second. The interface is an USB certified [LWS07] (represented in Figure 27 b).

Logitech QuickCam® Messenger – The QuickCam® Messenger (Figure 27 a) is assembled with a CMOS-Sensor. The resolution for video capturing as well as for still image is possible up to 640x480 pixels. The maximal frames per second rate is 30 and the interface equipped is a USB 2.0 certified [LWS07].



Figure 27: Cameras comprising the video system. a) Logitech QuickCam® Communicate™ STX Plus b) Logitech QuickCam® Messenger c) Creative Live! Cam Optia AF.

The way these three cameras are designated throughout this work are CAM1, CAM2 and CAM3 respectively. The reason for having chosen these three different cameras has the answer in the system design. These three cameras have two concrete characteristics that define them in groups of two, the brand (two Logitech and one Creative) and the video resolution (two with 640x480 pixel and the other one with 1600x1200). The first question is important to consider since cameras with the same brand, have (usually) the same drivers and this entails conflicts with the operative system. It seems obvious that having three cameras and two different brands the conflict will appear, therefore, to solve the problem, some measures were taken during the system implementation (they are explained later in section 3.1). The reason of choosing cameras with different video resolution is because the demand of image definition of some implemented image processing functions is different. Actually the camera entrusted to carry out more accurate detections (CAM1) is the one which has to cover greater distances (see Figure 25). Considering the dimensions of the SmaKi, CAM1 should achieve detections at 7,5 metres (approximately) distance, therefore is the camera which provides greater image resolution.

One feature common to these three cameras is the interface they compose. These three cameras compose an USB interface what make their use much easier; the plug-and-play capabilities are more and better developed for the current operative systems. High-speed serial buses such as the IEEE 1394 and USB 2.0 – or even USB 1.0 – are capable of transferring hundreds of megabits per second, a rate that greatly achieves the requirements of any common high-resolution video camera [AIP07]. Furthermore, although OpenCV supports many different camera variants like PCI, video grabbers or iee1394 (commonly known as FireWire), the one used most with OpenCV are the USB cameras [OCL07]. This fact derives in the greater reliability and variety for the functions and complements available for cameras composing such interface.

Chapter 3 System Design

The framework described in Chapter 2 in which this work is placed, involves a very specific procedure to follow in the system design. Once the ARS bases are defined, the procedure followed for a complete integration, is to evaluate the different necessities for the different purpose applications. The aim of this chapter is, firstly to describe what applications are wanted to be achieved with the integration of the computer-vision system in the ARS-PC (section 3.1). As next step are defined the different tools needed and the different variants available to carry out the computer-vision task, which essentially is the content of section 3.3. Finally, to have success in a full system integration are necessary those components that connect both systems ARS-PC and ARS-PC Vision, and make they able to communicate as if they were the same module.

3.1 Applications

The applications represent the final result of the whole work, which means what the system is capable to do once it is properly integrated. The developed system, since it is based on the ARS principles, is designed for building automation, therefore is intended to be applied not only in domotics (home automation) but also in offices [Pra06] in the endeavour to recognize human activities inside the buildings. The applications defined in this work are focused to prove the concept that the integration of new sensors – in the case in matter, cameras – with the other sensors – sensors of different nature – provide an information much more consistent and reliable of the environment. For the ARS-PC Vision were defined three different applications to accomplish different tasks, all three involve the identification of persons and are developed by means of the different symbols with support of the computer vision.

3.1.1 Person Tracking

“Person tracking” refers to the capability of the system to identify persons and be able to track them, in other words, to know their position during the stay in the room. It is an important fact to consider how many persons are in the room, as well as their relative position

to other objects of interest in the SmaKi, since is there where the different defined scenarios can occur (what is part of the application “scenario recognition” explained in section 3.1.2).

As starting point for this application, it was already mentioned in section 2.1.5 the consideration of a symbol Person in the ARS-PC with the necessary properties to achieve the task in which this application is involved. Furthermore the ARS-PC composes nowadays a robust system which achieves with a great grade of success the person tracking, by means of several functions, estimation methods, and the proper interpretation of the assembled sensors [Goe06,Ric07]. Like the symbol Person, the “person tracking” has a great importance since it is the base for the rest of the applications, not only for those which are considered in this work, but also in any other further applications that were considered in the SmaKi.

3.1.2 Scenario Recognition

In modern building automation is not enough that systems achieve a representation of the surrounding world. The entire symbol tree explained in section 2.1 tries to compose a symbolic representation of the world, but their purpose is to compose the base to identify different scenarios [Pra06]. Pratl defines scenarios as “*sequences of events subject to time constraints*”, but actually this definition seems to be too general to explain the application of this term through the development of this work. A suitable definition to be concreter taking into account the environment in which this work is focused, is the one given by [Cro05] that says that a scenario is “*a network of situations for modelling human activity expressed in terms of relations between entities playing roles*”. Considering this two definitions, the meaning of the concept scenario recognition applied to this work, could be resumed as human activity tracking, in which the previous mentioned entities are represented by persons and objects [Goe06].

Extrapolating this last definition to the ARS concept, is deduced that the symbol-level affected is the representation-symbol level, the one which the respective symbols Person and Object belong to. Therefore the way the computer-vision provides support to the generation of the different scenarios is by recognizing specific human features that detect those persons in the SmaKi, as well as by identifying different characteristics of concrete objects that are full of interest. The scenarios in which this work is focused are those depicted in Figure 28 which are “Person takes magazine”, “Adult makes coffee”, “Child makes coffee” and “Child near hot stove”. The first one is a complete new scenario in the ARS-PC symbol architecture, since the rest of the scenarios mentioned were already considered in the ARS-PC as they were well described in section 2.1.4. For these three last scenarios the task of the computer vision represents a merely support function in order to build a more robust system in matter of detection. The last two mentioned scenarios involve the recognition of children, intended to recognize dangerous situations, which is task of the application “Child safety” explained in the next section.

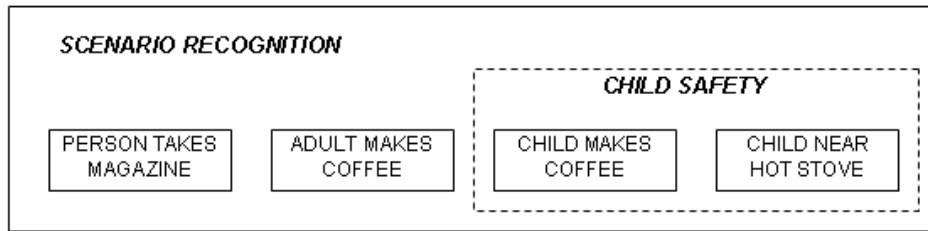


Figure 28: Scenarios involved in the “Scenario Recognition” application

3.1.3 Child safety

The third and last application for what this work is designed is the “Child safety”. One more time this application becomes very interesting considering the environment in which this work is focused, the SmaKi. It seems obvious that the risks for a child are less in an office than in a kitchen, where an adults negligence, like forgetting the stove working with a child inside the room, represents a clearly situation of danger for the child.

In order to avoid this hazardous situation, the application “Child safety” is considered as the one entrusted to take care of persons that were recognized as children, as well as to generate the corresponding scenarios involved in this matter when these situations of danger are recognized, which are “Child makes coffee” and “Child near hot stove”. Hence, as it is shown in Figure 28, this application results a specification of the scenario recognition explained in the previous section.

3.2 Running Modes

In Figure 26 within section 2.2.3 is represented that the information generated by the sensors is provided to the ARS-PC by means of the Octopus, nevertheless the system has the possibility to get old sensor data from the Sensor Database which represents a great advantage in the development of new software. With the implementation of the Sensor Database is possible to reproduce indefinitely stored scenarios, so long as the sensor data remains in the database, thereby the new developed applications can always be tested without the necessity of presence in the SmaKi.

To take advantage of this situation, the ARS-PC system was provided with two different program applications or “running modes” as part of the work carried out in [Ric07]. These two applications were designated *ARS_Live* and *ARS_Database*. The first running mode involves operation in real-time in the SmaKi, therefore the sensor information is got from the Octopus; since the second one gets the sensor data from the Sensor Database in order to generate the recorded scenarios. Hence, depending on what configuration is being used, the

modules in play are different; a graphical representation of this fact is given in Figure 29, where the figure on the left represents those modules that take part when the ARS is running in the ARS_Database configuration and the right one in the equivalent real time configuration called ARS_Live.

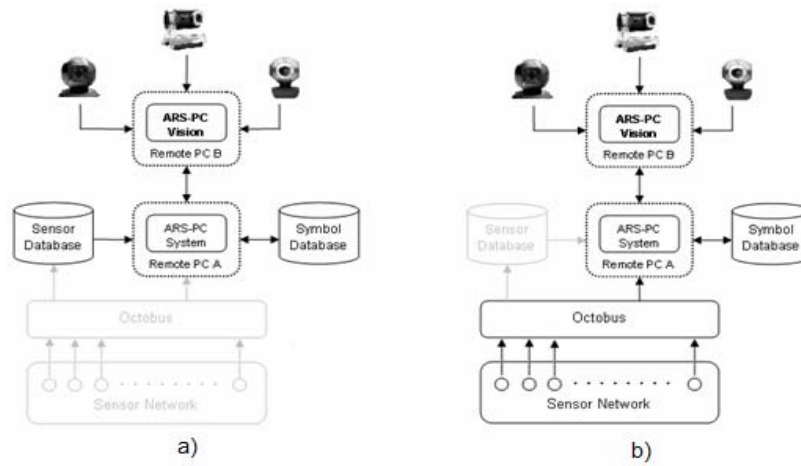


Figure 29: Modules in play in the ARS-PC. a) Case ARS_Database. b) Case ARS_Live

Considering these two operation modes in the ARS-PC, for a greater integration level, the ARS-PC Vision should be provided with two running modes analogous to the other two mentioned before, and another one additional intended to record scenarios. An important question to consider in the development of these operation modes, is that in the case of the ARS-PC Vision, all the incoming information from the sensors (except for the cameras) arrives through the ARS-PC, what means that the ARS-PC Vision does not really distinguish when the system is running a real-time or a database application. Therefore some synchronization methods should be developed, to have consistent information combined between the two sources of information, cameras and sensors.

3.3 Image Processing

This is the part of the work which represents a complete new feature in the ARS-PC, the one that concerns directly to the computer-vision field. By means of a computer-vision system, the SmaKi is provided with “sight sense”. In this section are described the different task in which the image processing must take part to have success in the recognition of the scenarios.

In section 2.3.2 it was mentioned that the libraries chosen for the image analysis was the OpenCV package. Furthermore two reason were given for having chosen it, one of them the great variety of functions and samples the package in matter contains. Those samples give a

great idea about what the OpenCV can do and how the different functions must be used. Since those samples have many interesting and useful integrated routines, they were sometimes used to develop bigger applications. In this section there are described the different image processing tools needed to develop the final applications mentioned previously.

3.3.1 Person Detection

The person detection is an important task to consider since it is involved in the process of all the different applications for what the system is focused. Previously the ARS-PC tried to identify a person with the sensors available in the SmaKi, which means with the floor sensors, with the motion detectors or through the activation of any from the other sensors. If a series of floor sensors are activated or the coffee machine is used, the system understands this as a human action, and the corresponding symbol Person is registered in the system. This way the system gives great results, but actually there are many other features that characterize persons univocally and make its detection much easier and reliable.

A common sample of these characteristics are the contrasts exhibited by a human face and their spatial relationships [OCL07]. The OpenCV package includes some classifiers – a classifier is anything that takes a feature set as an input and produces a class label [FP02] – to take advantage of these features, the so called *haarcascade classifiers*. These are some files that contain encoded patterns about Haar-like¹ features that consider the existence of oriented contrasts among regions in the image depending on the object that is considered. The OpenCV libraries include classifiers for recognizing faces and body shapes in different variants, full body, lower body, upper body, etc., as well as specific functions to apply those classifiers with their corresponding configuration [ORM01]. One sample included in the OpenCV package is the so called *facedetect*. This sample includes some routines for loading static images as well as series of images (video applications), and the subsequently application of the corresponding classifiers for the face detection. In Figure 30 is given a demonstration of this sample in which the program looks for the mentioned pattern, in order to detect the faces in the image. The depicted result shows some limitations of this program. In the image are six faces recognizable, nevertheless there are only four detected. The face in the background is not detected due to the excessive occlusion, and the other – the girl at the front – although it is clearly recognizable (there is no occlusion, eyes and mouth are clearly defined too) is not detected either.

1. The Haar-like features are so called because they share similarities with the Haar wavelets.

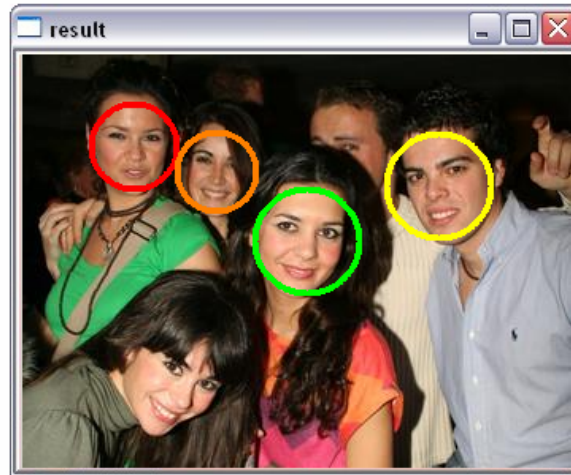


Figure 30: *facedetect* sample demonstration.

For the development of this task, considering the already given sample supposes a great simplification in sense of developing a person detection method; nevertheless it is important to take into account that the way the cameras are mounted provide certain advantages (or disadvantages) depending on what method is applied. These two considerations will be the starting point for the implementation of this task, described in the next chapter (in section 4.2.2).

3.3.2 Shape Distinction

The shape recognition is one of the greatest advantages of using cameras to recognize features from a specific environment or scenario. The term “shape” in this work has the same meaning that the one given by [Sin02], who refers to shape as “*the invariant geometrical properties among a set of special features of an object*”.

A similar problem, as the one explained in the previous section, occurs when a specific object wants to be recognized. Before the development of this work the ARS-PC recognized a new object in the SmaKi with no more information that the one given through the activation of the tactile floor sensors. That means that if the object in matter is on the table, on the bookshelf or anywhere where the activation of the floor sensors is not involved, the object will not be recognized. Even if an object is detected, the system is not able to recognize what it is, actually a person is recognized as an object until he/she makes some movements (or specific defined actions) around the room.

The shape distinction, integrated in specific visual recognition functions, not only composes part of the basis for the application “Scenario Recognition”, as it provides great support in the detection of objects, it also represents a very useful tool to accomplish other tasks, as it could be the possible detection of specific contours in the room in order to set points of reference,

what is for example necessary in processes like the camera calibration (explained in section 3.3.4) .

Most of the components in an environment such as the SmaKi, describe shapes that can be approximated to ellipses or rectangles. Taking this as a premise it is important the development of functions that achieve the recognition of such kind of shapes, regardless of the final application of those functions. Since the shape detection is a very common task in image processing [Sin02], the OpenCv package includes specific methods comprising the detection of polygonal contours: squares, trapezoids, shapes with multiple variants and even circles, detectable with the proper adjust of some parameters [ORM01]. Of course the image must be treated with additional processes, like segmentation or noise filtering [Sin02], to achieve a proper detection of the desired shape. Furthermore this software package contains one sample called *squares* that uses the mentioned function to find rectangles in static images with great results.

3.3.3 Background Subtraction

A fundamental task in many computer vision applications is the detection of objects in movement; changing pixels in an image provide an important feature for object detection and recognition [SS02]. Motion is a powerful feature of image sequences, revealing the dynamics of scenes by relating spatial features to temporal changes [JH00]. Considering the environment in which this work is focused, this kind of detection not only means a great support to the other sensors in the SmaKi in the person tracking task – the aim of tracking is to automatically find the same object in an adjacent frame from a video sequence once it is initialized [CY05] – but also involves a great tool in the recognition of different scenarios. To carry out this task, a common procedure is the background subtraction. This technique involves an observed image with an estimate of the image if it contained no objects of interest. The regions of the image where there is a significant difference between the observed and the estimated images indicate the placement of the object of interest. The designation “background subtraction” derives from the mere technique of subtracting the observed image from the estimated image and thresholding the result to generate the objects of interest.

There are many procedures to develop the background subtraction, starting with a simple difference between images [SS02], until using more complex algorithms like the Lukas-Kanade method that detects the optical flow considering the variation of the pixels intensity [LK81]. The background subtraction is a procedure that entails many challenges, like the light variation or the miss detection of unwanted non-stationary objects in the image.

The OpenCv package includes some samples that involve the motion detection by means of background subtraction using different procedures. The sample called *motempl* includes several routines to detect optical flow in a video sequence, but in essence uses a differentiation method between images with a specific threshold. Another sample is the one

called *lkdemo* which applies a variant of the Lukas-Kanade algorithm to detect motion in a video sequence by inserting in the image points that respond to the guidelines set by the algorithm in matter. Figure 31 shows the procedure of this demo-sample in which the program inserts several points (called later as *LK-points*) in the image contours that subsequently follow the optical flow from the corresponding points in the image.

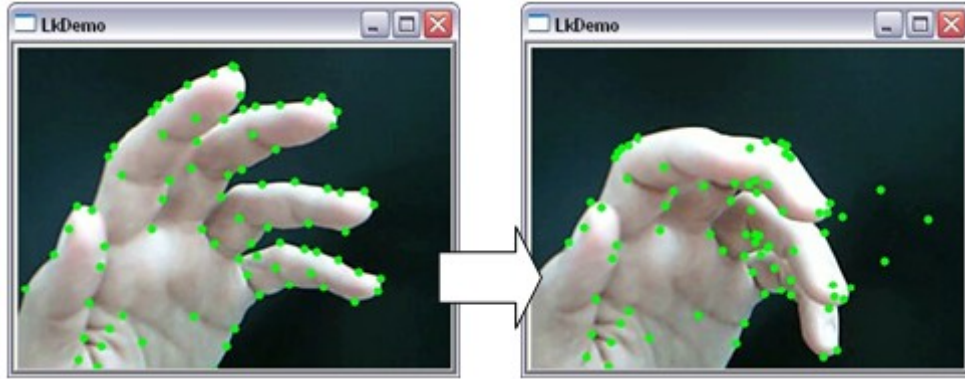
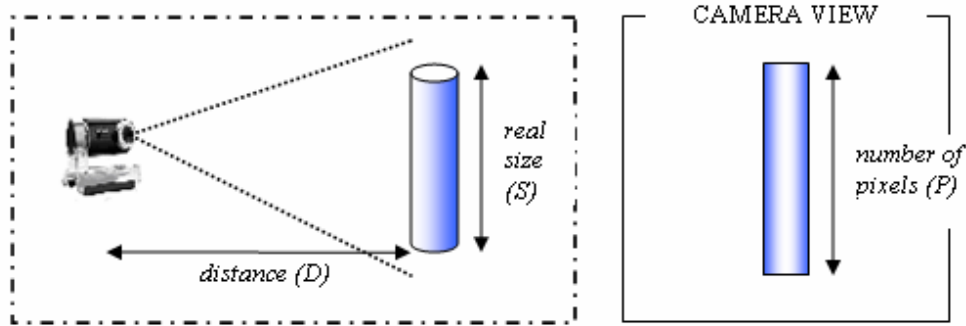


Figure 31: *lkdemo* sample demonstration.

3.3.4 Camera Calibration and SPD formula

Computer vision used in person tracking is normally carried out by means of two or more than two cameras. The reason of this fact resides in the mathematical incapacity that supposes to estimate sizes and distances with an only 2D view [Zan98, FP02]. The three-dimensional view is generated since two cameras are properly calibrated, which allows the system to get depth perception through triangulation. Hence, it is important to remark, that the concept of camera calibration used throughout this work is not the one intended to create a 3D view explained before, but to configure the camera coordinates and focusing in order accomplish properly the application in matter.

It was described in section 2.3.3 that three cameras were provided by the ICT for the development of this work, but due to the purposes set in the different applications, it was decided to use each camera focusing to different regions of interest without any synchronization among them, in sense of generating the mentioned 3D view. This decision leaves the system with three independent 2D views and the problem of the distances reappears. To solve it a mathematical relation among pixels, real dimensions and distances was experimentally deduced and subsequently designated as *SPD formula* (Size Pixel Distance). This formula shows that the real size of a concrete object is directly proportional to the multiplication from the distance camera-object with the number of pixels from the object in the camera image (see Figure 32). Furthermore the relation among these three variables results in a constant that is calculated taking as reference the same three variables for another object which values were previously measured.



$$\frac{S}{D \cdot P} = 2,5 \cdot \frac{S_{ref}}{D_{ref} \cdot P_{ref}} = cnt.$$

Figure 32: *SPD formula sketch.*

The number of pixels in image from the object in matter is normally a known value, especially when the whole shape of the object is clearly defined in the image. That means that in the *SPD formula* two unknowns remain, the real size of the object and the distance between the camera and that object, it is in the calculation of this two variables where the camera calibration is necessary. There are two possible ways to solve this problem, actually depending on the unknown desired to be resolved. If the searched variable is the size of a concrete object, considering the proper integration of the cameras in the SmaKi, it is possible to estimate the distance with the information provided by the tactile floor sensors, which give their position in the room each time they are activated (see section 2.2.2). If the position of the camera in the room is a known value (which is part of the calibration task), the distance between the camera and the object is resolved with a simply coordinates adjust. Consequently with the proper use of the *SPD formula* the value of the remaining variable is not an unknown anymore. Analogously, it is possible to calculate the distance between the camera and the object if the real size from the object is known. With this idea, if the shape of an element can be recognized and its size is a known value, the distance in matter is directly calculated with the *SPD formula*. If the object used for this last operation is fixed in a known position in the SmaKi, with the operation mode explained previously, the position of the camera in the room is deduced, actually a calibration from the camera is made.

3.3.5 Image Aberrations

As it was discussed in the previous section, in practice, the world and camera coordinate systems are related by a set of physical parameters, such as the size of the pixels, the position of the principal point, the position and orientation of the camera (tasks composing the camera calibration) and furthermore the focal length of the lens. Until know it was assumed that

cameras are equipped with ideal lens, however all cameras suffer from a number of intrinsic aberrations that must be taken into account in order to design a consistent system.

Radial distortion – A type of an aberration that depends on the distance between the imaged point and the optical axis. This effect is represented in Figure 33, where the figure on the left represents the ideal camera geometry and the one on right is the one resulting with the radial distortion effect. Geometrically, this distortion increases the distance between the image centre (c), and the point p – i.e. the displacement increases with the object height as the rays become more inclined. It is also deduced from figure in matter that the radial distortion does not an affect to the direction of the vector joining these two points.

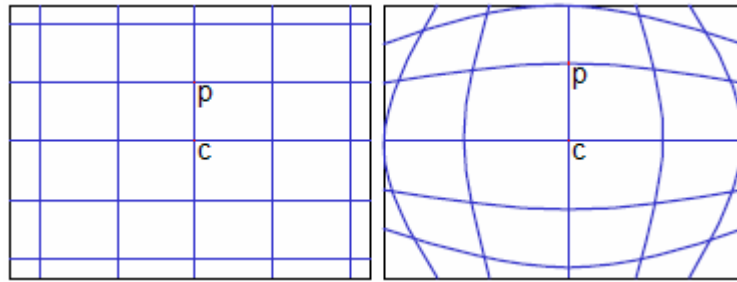


Figure 33: (Left) Ideal camera geometry (Right) Camera geometry affected by radial distortion

Chromatic distortion – Most of the aberrations related with the camera lens are caused of monochromatic aberrations which are caused by the nonlinearity of the law of refraction. The change of refractive index with wavelength causes polychromatic aberrations [JH00] – i.e. rays of different colours are bent differently by a lens. Therefore, the same scene spot is different for light of different wavelengths. For example, the image of a very sharp black-white edge may derive in a blurring or ramp of intensity change spread over several pixels in the image [HS92].

As the two mentioned aberrations there are many others due to physical phenomena concerning to the ray lights and the non-ideal camera components. For example, the spherical aberration, coma and astigmatism, which cause image degradations by blurring [JH00], but their origin as well as their solution are out of the scope of this thesis. Actually, although the importance of the mentioned aberrations, the one designated as radial distortion is the only one considered throughout this work (application in 4.2.3), due to the adverse effects introduced in some of the calculations carried out during the implementation.

3.3.6 Image Treating

Depending on the end application an image treating or pre-processing in the spatial domain is an important step to make the results more suitable for classification than were the original data. It eliminates different type of noise, sharpens the image features, such edges or boundaries or simply modifies its morphology to obtain concrete results. This process does

not increase the inherent information contained in the image, however it increases the dynamic range of the intrinsic features.

Histogram modification – The histogram of an image represents the relative occurrences of the different grey levels in the image. Histogram modification procedure modifies an image so that its histogram has a desired shape. This technique is used to enhance the image contrast and is a necessary condition for the subsequently application of other image treating methods.

Smoothing – Techniques which compose neighbourhood processing – in contrast to pixel processing – use the local context information for specific pixel processing. Commonly pixels close to each other have approximately the same grey levels, as exception for those which compose boundaries. The smoothing technique is commonly used to remove a kind of noise known as “salt-and-pepper noise”; it is a fine noise, produced by usually isolated points. This means that each point has non-noise neighbours [Sin02]. Hence the natural attempt is to reduce the effect of this noise by replacing each pixel with a weighted average of its neighbours.

Dilation – Is a morphological transformation – or process that affects do the geometrical structure within the image – that is used to remove very small group of pixels and join up groups that are closed together. Using dilation large groups can be joined up by “thickening” their boundaries. Figure 34 shows an example of the dilation procedure, in which gaps are jumped by attaching 3x3 neighbourhoods to each pixel; the relevant pixels are greyed on the right.

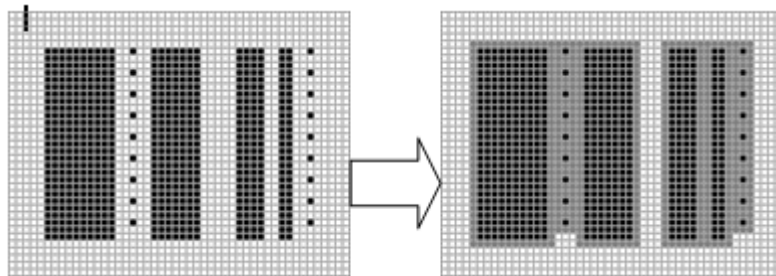


Figure 34: Representation of the dilation technique (Left) Original image.
(Right) Image with dilation effect.

Erode – The function Erode – also known as Erosion – like the dilation, represents a morphological transformation, however, the function in matter is intended to remove very small groups of pixels. This is carried out when a block of pixels do not fit inside a small group. In Figure 35 is illustrated the erosion procedure in which the small groups of dark pixels are lighted (greyed in this case) when they do not lie at the centre of a 3x3 dark neighbourhood.

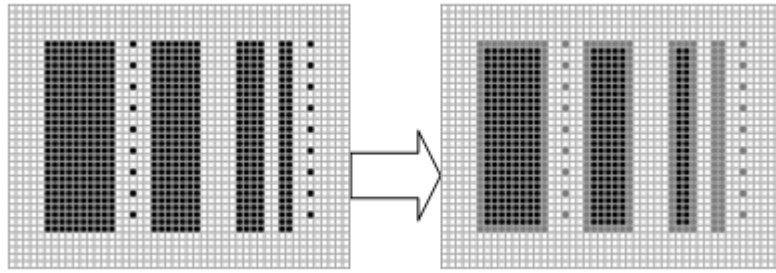


Figure 35: Representation of the erode technique. (Left) Original Image. (Right) Image with erode effect.

Frequently, these two operations (dilation and erode) are used together, either dilation of an image followed by erosion of the dilated result, or erosion of an image followed by dilation of the eroded result. With this procedure it is possible to eliminate some kind of noise or specific image details that are smaller than the structuring element – e.g. notches, blobs, gaps, etc. – while maintaining the essential geometric shape of the image. The combination of these two functions have a result similar to the smoothing effect [Sin02] .

Segmentation – This technique refers to the procedure of partitioning an image into multiple homogeneous regions; homogeneous in uniform function with some property, such as colour, intensity or texture. The resulting segmented image is a set of regions that cover the entire image. A particular case of segmentation is the technique called *thresholding* where individual pixels in a greyscale are showed if their value is greater than a specific threshold value or ignored otherwise. Figure 36 shows the result of applying thresholding with different levels of threshold; the one in the centre with a value of 70¹ recognizes the black object, those with less intensity. The one on the right with an almost three times larger the previous threshold identifies objects with greater intensity, the circle on the right and the small coin from the left.

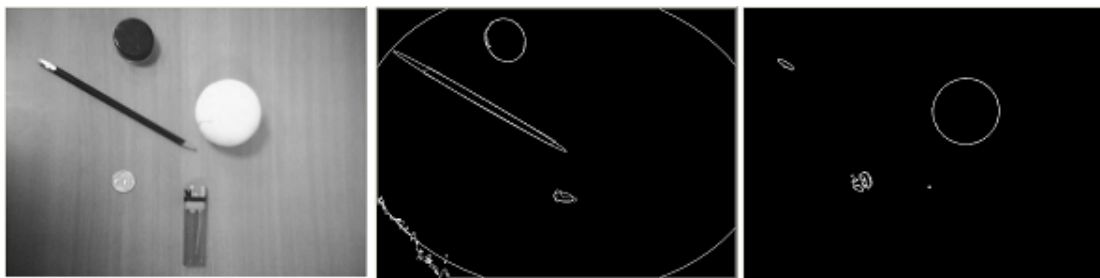


Figure 36: Application of contours tracing combined with thresholding in different levels. (Left) Original Image. (Centre) Threshold value equal 70 (Right) Threshold value equal 191.

1. This value is given in a scale of 0 to 255. Resulting pictures obtained using the *fitellipse* demo, sample composing the OpenCV package.

Edge detection and Contour Tracing – Strongly related with the shape distinction described in section 3.3.2, the edge detection is an important technique in the image treating and the scene analysis. “*An edge in an image is a contour across which the brightness of the image changes abruptly in magnitude or in the rate of change magnitude*” [RW96]. The goal of contour tracing follows another procedure as the edge detection, in which the boundary is traced by properly ordering successive edge points. A boundary can be viewed as a path formed by connecting the edge elements together. Thereby the edge pixels will represent worthier information that can characterize the shape of an object and its geometric features.

3.4 Interconnectors

Making analogy with the human senses, it is obvious that a greater and more accurate knowledge of the environment is achieved by means of the proper contrast among the five senses. Extrapolating this idea to the SmaKi, for a grater control about what happens inside it, the different sensors composing the SmaKi, tactile floor sensors, cameras and microphones (considered in other works [Pra06]), among others, should have full communication among them.

Nowadays, all the information provided by the sensors, with the exception of those generated by cameras, is managed by means of the ARS-PC and the language in which the entire code is written is Java. All the symbolic processing modules, the graphic interfaces, even the primitive version of the SymbolNet are developed in Java, but the OpenCV package is written in C and C++. That means that the whole vision system needs its own code, in the sense of symbolic processing modules, a proper SymbolNet version as well as the necessary interconnectors in the endeavour for a complete system integration.

The interconnectors that compose the ARS-PC Vision are divided in this section in two parts, those which are necessary for the symbolic data exchange, like a specific sender and the corresponding receiver (section 3.4.1), and others that support the assembly and disassembly of symbolic packages (section 3.4.2).

3.4.1 Two-way connection

Currently the ARS-PC system is composed of some classes that make the network connection with all the necessary modules, like the Octobus, the SmaKi Visualization module (developed in [Sch07]), and of course the ARS-PC Vision. These classes, TcpSender and TcpReceiver [Hol06], are part of the SymbolNet package and as it is obvious, they use the TCP network protocol to send and receive information between different symbolic processing modules. Moreover, to accomplish the connection establishment in both ways those functions implement a specific handshake in which the version of the SymbolNet used from the two parts in the connection is proved. Hence to develop a consistent system, the functions

implemented in the ARS-PC Vision are designed with the same specifications explained previously.

Once the connection is established, the ARS-PC sends constantly symbolic data packages in DER [Dub00] encoded, as soon as they are generated. That means that the receiver in the ARS-PC Vision must compose a procedure that takes into account this constant data flow and store it properly for the corresponding decoding and treatment. Analogously the ARS-PC Vision needs a procedure to assemble data in the corresponding symbolic packages (explained in the next section), as well as a method to encode those packages properly and send them through the network.

3.4.2 Symbolic Processing

As explained in previous sections, the communication among the modules in the ARS-PC is made by means of symbolic data packages. Depending on the type of message (NSM, UPM or EM see section 2.2.4) the packages contain different fields, but all of them maintain the SymbolNet structure.

Once the ARS-PC Vision receives one message and is successfully decoded, some procedures are necessary to check what kind of message is received, as well as some methods to get the wanted information within the incoming messages. The first one designated in this work as *proof functions*, can be developed with concrete knowledge from the SymbolNet structure, by looking inside of the proper fields from the message. The other functions are designated later as *getValues* functions, are iterative routines that look for a concrete value in a symbolic message and facilitate it for the further use of other functions.

Furthermore, the ARS-PC Vision must implement some functions to achieve the inverse process, which composes some methods that achieve the assembly of new messages containing the important results obtained in there, for the subsequently sending in ARS-PC direction. To build a consistent system the symbolic messages should be those from the SymbolNet package explained in section 2.2.4, composing the respective fields depending on the kind of message and considering the already existing symbols and the properties that compose them, which are briefly enumerated in section 2.1. The kind of symbols sent by the cameras should always be from the microsymbol level, as the rest of the data source devices do [Pra06]. The camera should be in the same level as the floor sensors or motion detectors or in any case, in the same level as all the sensors considered in the system, which is clearly depicted in Figure 20 as the sensor layer.

Chapter 4 System Integration

In Chapter 3 the different purposes and necessities for the aim of this work were described as part of the system design. Making analogy with the previous chapter, the ARS-PC Vision system was divided in three packages to accomplish the different tasks the system was designed for. In Figure 37 is shown the resulting sketch of these three packages and enumerated the different modules that each one contains enumerated.

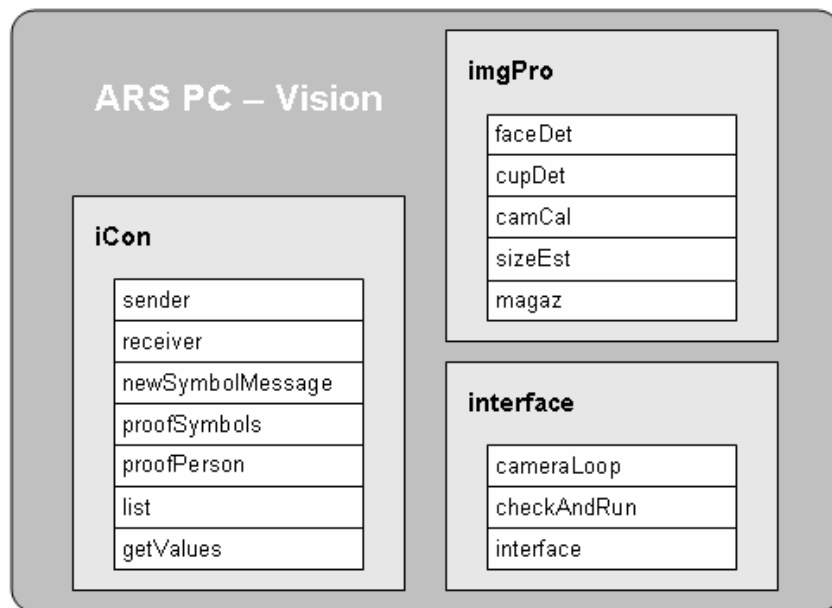


Figure 37: ARS PC-Vision modules distribution

These three packages and the section belonging to the system applications are described in this chapter following the bottom-top principle. First of all, to accomplish the integration between ARS-PC and ARS-PC Vision the package designated as **iCon** (interconnectors) is described in section 4.1 together with the modules that carry out the connection between the two systems and the necessary symbolic processing for the proper management of the information. After that, in section 4.2 is described the **imgPro** (image processing) package, which contains all the modules designated for the computer vision tasks. Finally, as the last

part of the integration, in section 4.3 is described the package interface which implements the necessary modules for the control of the program.

4.1 Interconnectors (iCon)

It was described in section 3.4 the necessity of some modules to achieve the proper interconnection among the already existing symbolic modules composing the ARS-PC and those which represent the network interface with the ARS-PC Vision. The iCon package contains all the modules that deal with symbols or symbolic information. This layer is entrusted to receive the information from the ARS-PC, make it accessible for the computer-vision layer and send the obtained results back to the ARS-PC with the proper format. All the functions composing each module in this package are based in the SymbolNet.

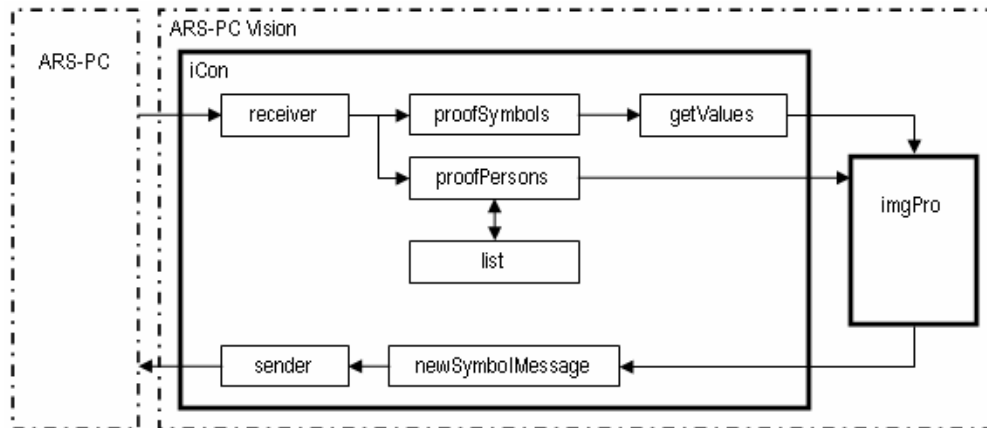


Figure 38: iCon package break down with system interconnections.

Figure 38 shows the iCon package with its composing modules and the corresponding connections among them. From this graph is deduced the placement of this package in the whole system, the connection with the ARS-PC system and the relation with the imgPro – or computer-vision layer – package (explained next in section 4.2). The iCon package represents the interface between the ARS-PC and the ARS-PC Vision, obtains the information generated in the ARS-PC and manages the data doing it intelligible for the rest of the system at hand. Consequently the inverse procedure is considered too, the information generated in the imgPro is formatted and sent back to the ARS-PC by means of the iCon package. The modules depicted in Figure 38 are briefly described next:

receiver – This module is entrusted with the network tasks. It is composed by a set of functions that treat directly with the ARS-PC and manages the constant data flow incoming in this direction.

proofSymbols – The module `proofSymbols` is composed by several functions, which are designed to check concrete symbols in order to prove its utility for the system.

proofPersons – Due to the importance of the symbol `Person` in the entire system, this module, as a specialization of the previously mentioned module, is only focused in verifying the symbol in matter. Furthermore it contains other methods to manage the information contained within the symbol `Person`.

list – In order to register the number of (symbol) persons in the system is implemented this module, which manages a list of such kind of symbols under the control of the module `proofPerson`.

getValues – As part of the process of managing the symbols, this module is composed by several functions that are designed in to extract concrete information specific for different symbols in order to make it intelligible to the other applications.

newSymbolMessage – This module is entrusted to do the inverse process as the one explained before, as it generates the new symbols with the information obtained through the computer-vision layer.

sender – The sender module takes part in the network parts, as the previous `receiver` module does. It is essentially designated to send each new symbol generated in the ARS-PC Vision in the ARS-PC direction. Together with the `receiver` package, this package represents the interface ARS-PC/ARS-PC Vision.

4.1.1 Sender

In the module called `sender` two functions to accomplish the function of a “SymbolNetTcpSender” are developed. The first one called `connectSender` establishes the TCP connection with the ARS-PC which is supposedly waiting for the connection coming from the ARS-PC Vision. The other function in this module, the one designated as `sendSymbol`, is the one entrusted to send the symbol-messages (see section 2.2.4) with the corresponding encode in the ARS-PC direction. Those messages are previously packaged with the proper SymbolNet structure by means of the `NewSymbolMessages` functions explained later in section 4.1.3.

4.1.2 Receiver

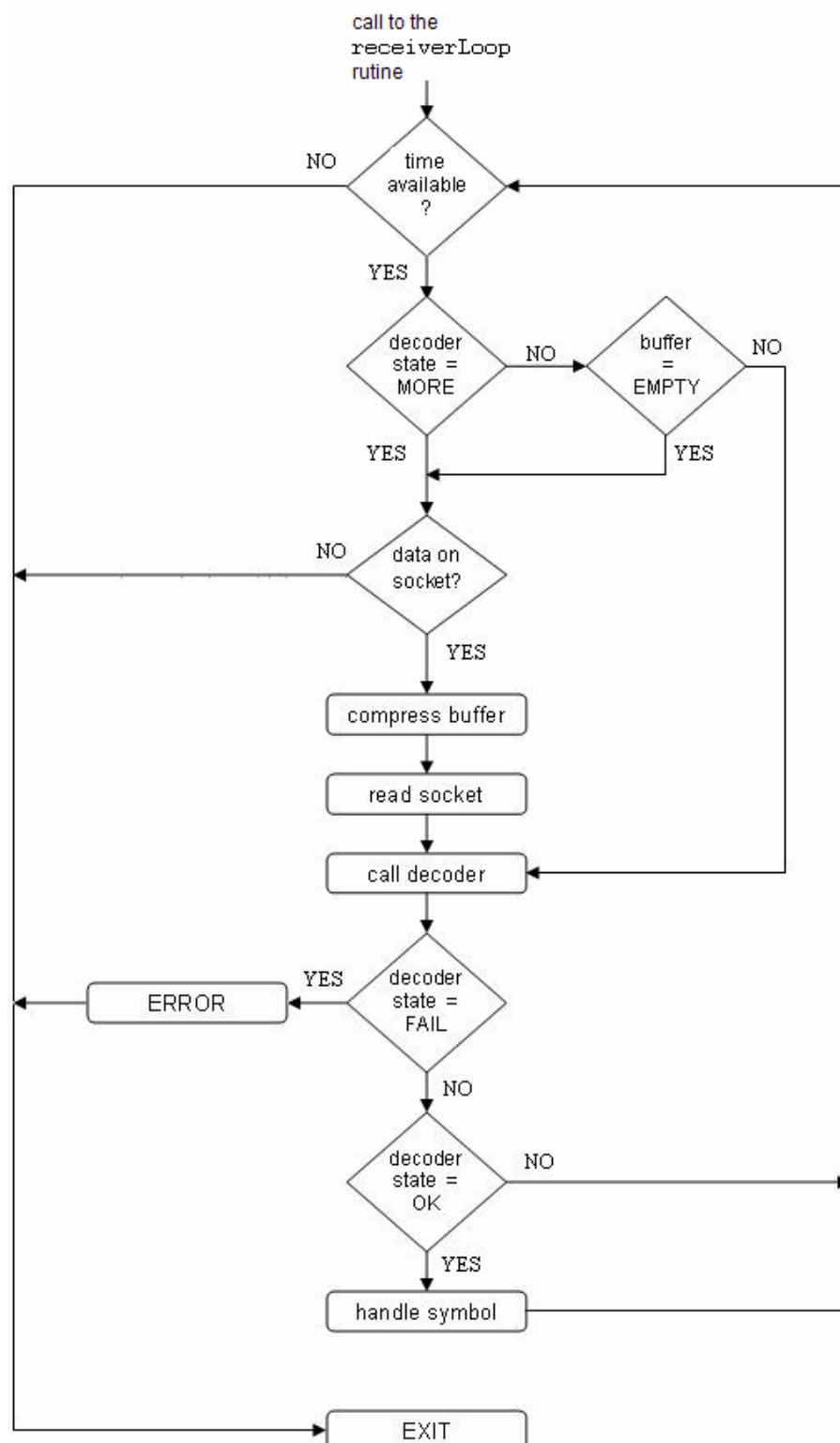
The receiver in the ARS-PC Vision is placed in the module with same name. The module `receiver` is composed of two functions, `connectReceiver` and `receiverLoop`. The first function makes a TCP connection, listens to a network port and waits for a connection from a “SymbolNetTcpSender” and the second one is entrusted to manage the incoming symbolic data flow. After the connection is successfully established, the function `receiverLoop` can be always called if it is necessary. Due to the kind of implementation

of the system, this function must be non-blocking, since the system must attend to other functions like the one which manages the video capturing (further explanation in section 4.3.3), therefore the period the system remains in the loop is controlled by time.

The function works mainly considering the state of three components, the decoder, the buffer and the socket, and remains in the loop until some specific conditions are fulfilled. The decoder can have three different states, *RC_OK*, which indicates that the decoding was successfully accomplished, *RC_WMORE*, means that the decoder expects more data, and *RC_FAIL*, which results when a failure occurred during data decoding. Moreover it is important to consider the state of the buffer, whether there is data in it or, on the contrary it is empty. And finally, after considering the previous variables, the program must check if there is any data in the socket or not. The different steps followed in the loop are shown in Figure 39 and described next:

Actually the permanence of the system in the receiver loop is controlled by means of two conditions. The first one is a temporary condition, which does not permit the system to remain in the loop more than a specific time given in milliseconds. The other condition is given by the importance of a specific symbol. In the case the system is waiting for a specific symbol, what happens for example at the beginning of the system in the *Database Mode* (see section 4.3.4), the program keeps looping until the symbol is received. On the other hand it is possible that the system must exit without waiting for the time condition to expire, because the information arrived must be immediately processed by another module.

If the conditions are the suitable to remain in the loop, the function looks if the decoder needs more data or if the buffer is empty. Whether one of both cases is affirmative, the system checks if the socket for the consequently reading and decoding. After the decoder accomplishes a successful decoding, the received message is treated by different functions – like those from the “proof symbols” group or the “get values” (explained in sections 4.1.4 and 4.1.7 respectively) – to manage the information incoming within the messages.

**Figure 39:** receiverLoop flowchart

4.1.3 NewSymbolMessage modules

As part of the system design was discussed in 3.4.2 the necessity of some concrete modules, specific for each kind of symbol generated, to make the corresponding packages with the symbolic information. In section 2.2.4 it was given a superficial idea about the SymbolNet, but the necessary features of this software-package to develop this work. The possible messages to use with this protocol were enumerated as well as the fields and properties that compose the structure of a symbol. In this section the possible messages the ARS-PC Vision is able to generate as the result of any kind of detection by means of the cameras, are described. The only question that differs from the proposed structure of a symbol is that the class given for each symbol sent by the ARS-PC Vision corresponds to the group of the sensors; each type of message from the ARS-PC Vision will be treated as a sensor value. This consideration was taken due to the great simplification it represent in order to integrate these symbols in the system. The different symbol-messages the ARS-PC Vision generates are described next:

nsmCup – The module entrusted to encapsulate in a NewSymbol-Message the important data related to the “cup detection” is the nsmCup. The main routine is the called nsmCup; it gives the entire structure for the symbol, with its different fields and properties, and calls the other routines to make sub-encapsulations. In this module there are two methods that build the property structure, for its subsequent addition to the symbol structure. These two functions are mpCup and mpIDcup (“make property Cup” and “make property ID cup”, respectively). Each one sets the suitable class and scheme for each property, and gets the values for these properties from other two functions, mpvCup and mpvIDcup (“make property value Cup and “make property value ID cup”). The first one sets the value concerning the status of the “cup detection”, information incoming from the module cupDet described in section 4.2.6. The other one inserts the sensor-ID designated to this symbol in the mpIDcup function. The diagram of the module nsmCup with its methods and connections among them is shown in Figure 40.

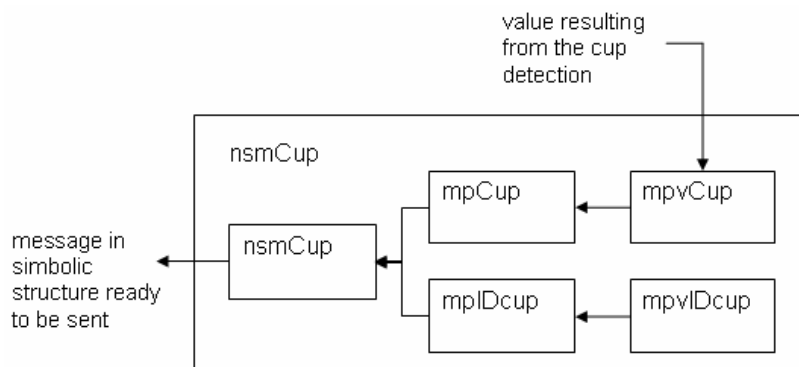


Figure 40: nsmCup module diagram.

nsmHeight – In the module `nsmHeight` is the *person-height* built as a NewSymbol-Message. This module follows the same pattern as the previous ones. In this case there are three different properties included in the symbol, which are “height”, “sensor-ID” and “position”, added with their consequent property structure by the functions `mpHeight`, `mpIDheight` and `mpPosition` respectively. The value for the property “height” is inserted by the function `mpvHeight` which takes the value directly from the “person height detection”, carried out by the `sizeEst` module (see section 4.2.3). The `mpvIDheight` method adds the “sensor-ID” which makes this symbol univocal among the other sensors (remember that symbols sent from ARS-PC Vision are considered each one as sensors, in the same level as the floor sensors or the contact switches). For the last property value in this symbol, there is the `mpvPosition` which sets the position where the height detection was carried out. This position is given through three values, which correspond to the SmaKi coordinates, width, depth and height. In Figure 41 is shown the structure of the module in matter, the interconnections, and the difference between the structure of the module `nsmCup` with the new property (position) added.

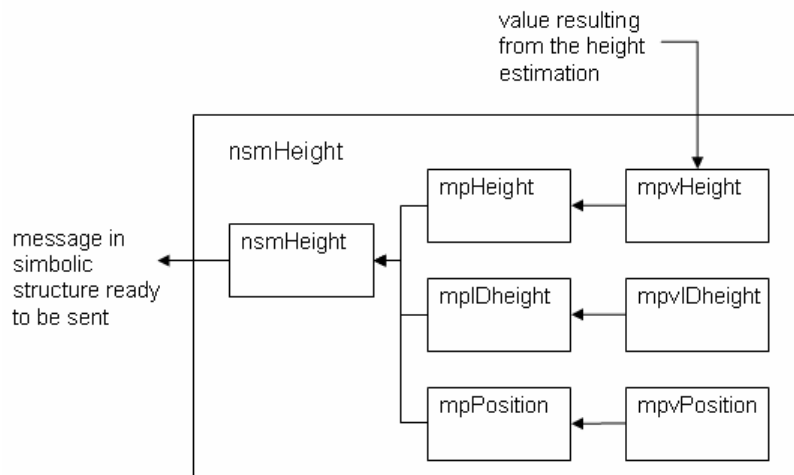


Figure 41: `nsmHeight` module diagram.

nsmMagaz – The module that generates the NewSymbol-Message with the necessary information from the “magazine control” is the one called `nsmMagaz`. Like in the other NewSymbol-Message modules, there is a main function `nsmMagaz`, which gives the characteristics which make the symbol univocal, like the symbol class and scheme. Furthermore, it assembles the different symbol properties in the symbol structure. In this symbol only two properties are added, the status of the bookshelf (if magazines are missing or not) and the one mandatory for all the symbols generated in the ARS-PC Vision, the “sensor-ID”. Hence there are two functions needed for building the symbol-property structure and other two give their respective property-values. These functions are `mpMagaz`,

mpIDmagaz, mpvMagaz and mpIDmagaz, with analogous connection to the previous modules.

4.1.4 Module: Proof Symbols

It was mentioned in section 2.2.4 that symbols are understood as data packages, since each symbol is composed by some fields and different properties. To recognize the incoming symbols, to distinguish one kind from the others, it is necessary to look into the fields composing each symbol (remember that each symbol and its property are strictly defined by respective classes). Following the guidelines described in section 3.4.2 the module `proofSymbol` contains different functions to recognize or “prove” what kind of symbol is being managed. In the most of the cases the task consists in looking for the class and the schema that defines univocally each symbol. All these functions return true or false, depending on whether the conditions established for each routine are accomplished or not.

proofRectangle – This routine is designed to check if the symbol in matter is a microsymbol object and if the property position appears in it. Since the property position in this symbol is given as a rectangle, the function `proofRectangle` proves if the field schema corresponds to a rectangle.

proofFootstep – The function `proofFootstep`, looks into the class of the managed symbol, checks if it corresponds to the class footstep and makes sure that the current symbol contains the property position given as a rectangle.

proofGait – This function checks if the symbol in matter is a symbol Gait and if the property “start position” (given with the schema point) appears in the symbol.

proofDoorStatus – Function `proofDoorStatus`, returns true if the treated symbol has the corresponding class to the microsymbol “Door Status”.

proofHBmsg – This function checks if the incoming symbol is a HeartBeat-Message. The application of this function is merely question of performance optimization. HeartBeat-Messages are sent constantly from ARS-PC to the ARS-PC Vision. Since the ARS-PC Vision (in principle) does not use such messages, the system in matter uses this function to ignore these messages directly when it is received.

All these functions, with the exception of `proofHBmsg`, check only the symbol when it appears as a NewSymbol-Message. If it is necessary to control the update of one of these symbols, it will be made by means of its ID. There is another function called `proofPerson` which has a similar purpose of the functions mentioned before, but due to its complexity it is necessary to place it in another module, accompanied with other routines. Of course “proof” functions can be made for each symbol but the different functions developed in this work were enough to fulfil the requirements of the system.

4.1.5 Module: Proof Person

The aim of this module, with the support of the module explained in the next section (module `list`), is to control the number of persons in the room and its relative position to different objects in the SmaKi. The `proofPerson` module contains two functions, `proofPerson` and `nearObj`, both strongly connected.

nearObj - The `nearObj` function is one of the “proof” functions since it checks if the managed symbol has the property *Near*. As explained in section 2.1.5 the property *near* appears with the symbol *person*, when this person is near to a concrete object in the SmaKi. The value of the property is univocal, depending on what object the person is near to. So the function `nearObj` will return true if the person is near to that object. This function considers two different kind of incoming messages, NSM and UPM.

When the function treats a NSM it looks inside the symbol for the class corresponding to property *near*, if it is found, its property value will be compared with the value which references to the wanted object. If it matches, the function will return true, the person will be considered near the object. In case the message in matter is an UPM two possibilities must be considered. Firstly, if the symbol (*person*), the UPM is sent for, was already considered as near the object, only if the property *near* appears within the symbol with another value means that the person is not near the object. Secondly if the property *near* does not appear, is because the person is still near the object. However in the case that the symbol treated, was not considered before executing the function as near the object, the routine looks for the property *near* and its current value in the case the property appears. The opposite case will occur if the person is not near the object. Of course for both cases, it can happen that the value obtained in the property value is not the one expected, which would mean that the person is near an unwanted object.

This function does not care about what class the symbol belongs to, since `nearObj` is called inside the `proofPerson` routine and despite this fact the property *near* can only appear within the symbol *person*.

proofPerson – Depending on the type of message received (NSM, UPM or ES) the function will react differently. If the incoming is a NSM, it is necessary to check to what class it belongs to. If the symbol is a representation-symbol *person*, since it is a NSM, the person will be considered new in the room and will be added to a list, where all the subsequent persons will be added to. The symbol is stored in the list with its ID and the information about its relative position to a concrete object (by means of function `nearObj`).

In case the received symbol is an UPM, the mentioned list will be searched through looking for a match between the ID from this last symbol and any other contained in the list. This match will mean that this message is an update for an existing symbol. With the help of function `nearObj` it will be checked if the person in matter is now near an object, still near

an object or gone from an object. After that the list will be updated with the new values concerning that person. If the message is an ES (see section 2.2.4) one, it will be proved if the ID contained in the message belongs to a person from the list. In the affirmative case, the person will be deleted from the list as well as the related information. All the functions related to the management of the list are contained in the module described next; module `list`.

4.1.6 Module: List

The module `list` is composed of three functions to manage the list which controls the quantity of persons in the room. It is designed to generate, delete and find persons in the list and manage the properties containing each symbol.

generateElement – The routine `generateElement` adds to the list a new person. The person is inserted in the list with its ID (symbol-ID) and with properties concerning to its relative position to a concrete object.

deleteElement – The function `deleteElement` looks into the list for a given ID, and in the case of match, the person with its respective properties is deleted.

findPersonInList – The function `findPersonInList` searches in the list for a given ID, and returns true if the ID is found. That means that the person with that ID already exists in the list, in other words, the person exists in the SmaKi.

This module is focused for the application person tracking as well as the recognition of scenarios, since it controls which persons are in the room (through its symbol-ID, explained in section 2.2.4), the number of persons and also registers the relative position to the fixed items in the SmaKi, such as the coffee machine, the fridge or the bookshelf. In this work this module is directly applied to the recognition of the scenario “Person takes magazine” explained later in section 4.4.3.

4.1.7 Module: Get Values

Once again it is important to remark that the symbols managed in the whole system are understood as data packages. Therefore it was already mentioned in section 3.4.2 the necessity of some functions to extract a concrete value within a specific symbol. Hence, the module `getValues` is designed to extract this information and make it useful for other applications. The operation mode is different depending on the type of message (NSM, UPM or ES) and the way each function extract the information may be different depending on what kind of data is desired to be obtained. Usually, if an incoming symbol is proved it is because some information from this symbol is needed or relevant. Hence routines from `getValues` module will be normally called after a function from the module `proofSymbols` is executed. The different routines included in this module are described next:

getTimestamp – As explained in section 2.2.4, all the symbols have a field in which the generation or update (in case of UPM) timestamp is contained. The function `getTimestamp` returns the timestamp of the corresponding symbol independently of what symbol class is treated.

getDoorStatus – For the right operation of the system, this function should be called after the `proofDoorStatus` function. The routine `getDoorStatus` looks into the properties of the symbol “Door status” and returns the current state of the door, if it is open or closed.

getPosGait – This function needs the results from routine `proofGait`. If the result given by `proofGait` is positive, `getPosGait` returns the value of the property “Position” (described in section 2.1.5) from symbol Gait. The symbol Gait represents the trace of one person in the room, and after its generation, it always reappears as a symbol-update. Due to this fact and considering that the symbol-class does not appear in the UPM, the only way to manage the right symbol after its generation is controlling its ID.

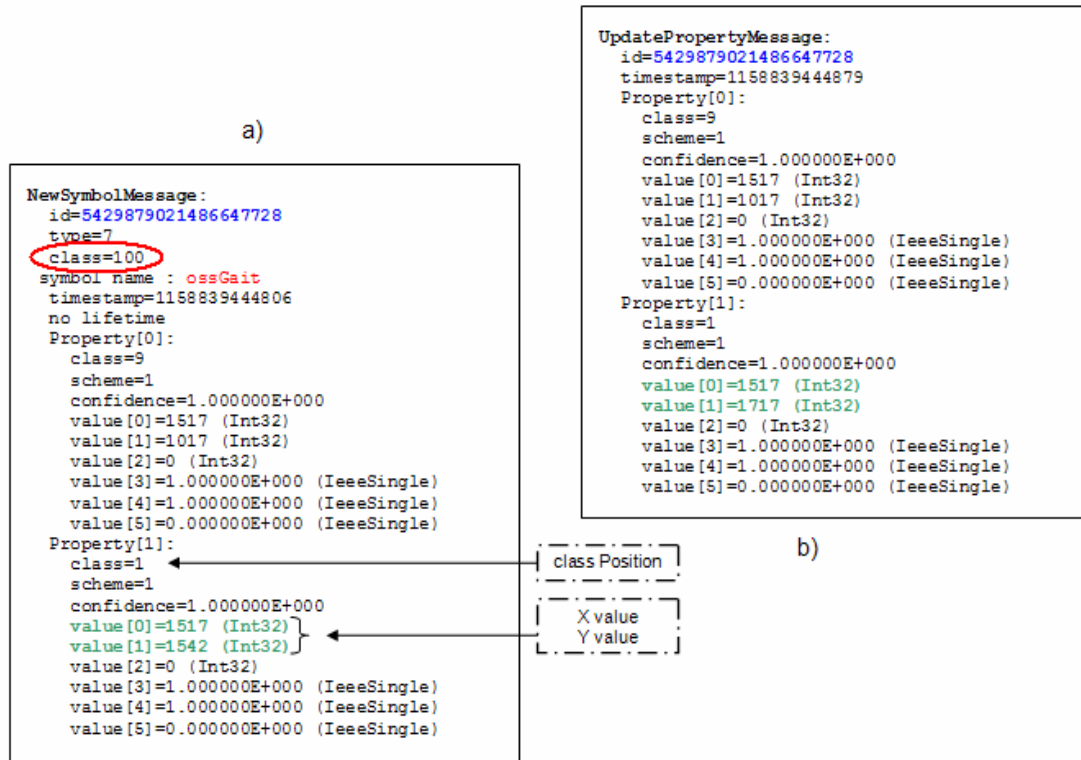


Figure 42: Outcome for incoming snapshot-symbol Gait: a) NewSymbol-Message, b) respective UpdateProperty-Message.

Figure 42 shows the outcome of printing an incoming NewSymbol-Message and the UpdateProperty-Message for the same symbol (Figure 42 a) and b) respectively). These both messages belong to the snapshot-symbol Gait, what is defined by its class (circled in red) in the case of the NSM and it is recognized later in the UPM by means of its ID (the same

symbol-ID in both messages, in both pictures, depicted in blue). The values wanted in this case are those which are contained in the second property, which corresponds to the property Position, given in two coordinates (actually in three, nevertheless the third one represents the height coordinate in the room always given with a null value). Hence the function `getPosGait` returns the first and the second value of the second property, which represents the current position, in x-y coordinates, where the gait was generated.

getPosObj – The function `getPosObj` returns the current position from a detected object. This value is the result of looking into the *microsymbol* object. The *microsymbol* object is directly generated by the information coming from the tactile floor sensors. Explained previously in section 2.2.2 is that those sensors give three points to define their position. Hence the property position concerning to that symbol is given with these three points.

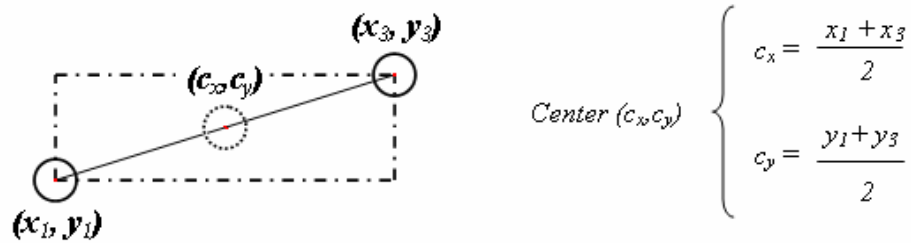


Figure 43: Calculation of the centre point within `getPosObj` function

The function `getPosObj` with the only information from two of these points, calculates the centre of the described square with a simply mathematical operation (see Figure 43), and returns this value as the current position from the object in the room. This function is supported by the `proofRectangle` function described in the previous section.

4.2 Image Processing (imgPro)

All functions entrusted to make any image processing or those entrusted to support this task are contained in this package and they represent the computer-vision layer.

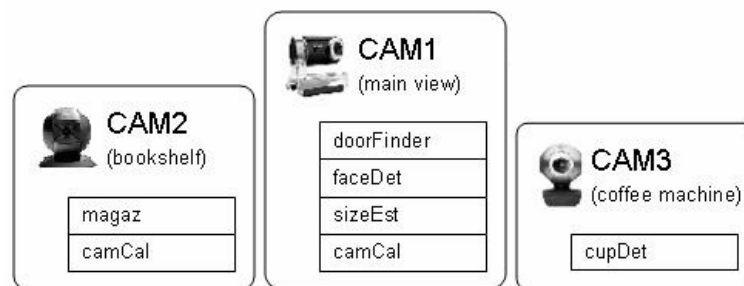


Figure 44: Modules-cameras correspondence

In section 2.3.3 it was mentioned the availability of three different cameras for the development of this work and in section 2.2.2 was described the placement and focusing for each camera in the room. Following the requirements for each application, each camera is assigned to different modules. These correspondences are shown in Figure 44 and briefly described next:

doorFinder – This module is intended to find the main door in the camera CAM1 view field in order to set an univocal reference in the room. Such reference is necessary to carry out other applications like the one designated for the camera calibration.

faceDet – The `faceDet` (“face detection”) module is entrusted to locate persons in the room. This is achieved by detecting faces in the camera CAM1 view field. Moreover it provides useful information to the operation of other applications designated to the same camera.

sizeEst – This package, which designation derives from “size estimation”, achieves the estimation of the height of a detected person. With this module it is possible to carry out the distinction among children and adults. Actually the rest of the modules designated for this camera (CAM1) are focused to provide necessary information to the procedure of this application.

camCal – The module `camCal` or “camera calibration” is a set of functions specific for each camera that needs some kind of calibration. Depending on the end-application designated for the camera, the calibration has different purposes and therefore it follows different procedures.

magaz – The designation “magaz” derives from “magazine”. This module is the one entrusted to do the necessary procedures to achieve the control of the bookshelf, necessary for the recognition of the scenario “Person takes magazine”, carried out by means of camera CAM2.

cupDet – The `cupDet` (“cup detection”) module is the one designed to detect cups (coffee cups) in the camera CAM3 view field. This package is developed in order to provide support to the recognition of those scenarios in which the act of making coffee is involved.

4.2.1 Door Finder

In the endeavour to set a reference for the camera calibration, a concrete object with univocal features in the SmaKi was selected. This is the case of the main door. Although the picture from Figure 45 does not correspond to the one from the SmaKi, this one has the same physical properties: the same dark blue colour, the same measures (87x200cm) and the same white background. No other object in the SmaKi has these properties and its placement in the room is privileged, centred on the left side of the room.

One interesting method to be used to identify the main door in the room is using colour histograms – a colour histogram is a record of the number of pixels in an image or a region that falls into particular quantization buckets in some colour space [FP02] - as it seems obvious in Figure 45, the colour from the door is unmistakable. But apart from some reasons that are explained next, it was mentioned in section 3.3.2 that the OpenCv package composes a great sample that is the *squares*, which tries to find squares in the image. Taking this sample as the starting point, the so called *doorFinder* was developed in the module called with the same name, as an application composed of several functions.

findDoor – Whenever the *doorFinder* is called by the system, the function `findDoor` manages the image incoming from the corresponding camera, obtains each isolated frame for each camera capture and creates the corresponding frame copy for the subsequently image processing carried out by the functions explained next.

findSquares – This function returns a sequence of the squares detected on the image. To achieve this task the function creates an empty sequence that will contain points, 4 points per square (the squares vertices).

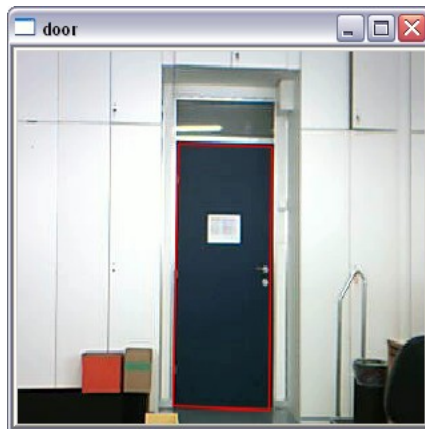


Figure 45: Door identified with the *doorFinder*

For the process explained next it is necessary to filter out the noise, which is made by means of a down-scale and a subsequently up-scale from the image. To optimize the search for the door, the COI (Channel of Interest) set for the image treated, which is an RGB one, is the third channel and corresponds to the colour blue. After that operation the system tries several threshold levels in which the contours are searched (it was already explained in section 3.3.6 the aim of the thresholding procedure which belongs to the techniques of segmentation). Any time the program finds a contour, it is checked to prove if all its angles are of approximately

1 In this work, in matter of detection, squares and rectangles are mentioned without distinction.

90 grades. This is the only condition a contour in this case must fulfil, to be stored in the sequence of squares, actually with that condition squares as well as rectangles¹ are detected.

drawSquares – Since the previous function finds all the possible rectangles in the image, the function `drawSquares` discriminate among all the contours stored in the sequence made by the function `findSquares` to find the one which corresponds to the door and draw the respective rectangle that defines it.

The procedure followed by the routine to recognize the door is carried out by means of its proportions. Since the door in the SmaKi has known proportions, the system will choose the rectangle in the sequence which proportions approach most to the door measures. As explained with the previous function, the squares are stored as 4-points sequences. With this information it is possible to determine the height and the width for each square, and hence its proportion (defined as height/width). When the square with the proper proportions is found, the respective square is subsequently drawn on the image where the whole process was carried out (see Figure 45).

4.2.2 Face Detect

In section 3.3.1 were mentioned different procedures to carry out the person detection by means of image analyzing. Due to the requirements of the system explained next, the one chosen for the development of this task is face detection. The profits of placing CAM1 (camera for which this application is assigned) focusing on the door are a larger vision field and greater control of the regions of interest, like proximities to the stove, coffee machine and even the door. However the way this camera is focused, leave regions in the room where the entire shape of a person would not fit in the camera view. That is the reason why the “face detect” was chosen instead of “body detection”. Furthermore the results from the face detection are optimum for the module `sizeEst` explained in section 4.2.3, which is used to estimate the height of a detected person. The module in which this task is developed is called `faceDet` and is composed of the functions described next:

faceDet – This function is the one called to execute this module. `faceDet` makes the corresponding image capture from the camera CAM1 and prepares the frame for the subsequent call of the function which makes the face detection, function `detectAndDraw`.

center_axes – This function draws the axes in the desired image centred in the middle point of the image (depicted in blue in Figure 46). Moreover return the coordinates of this point in the image. Although the simplicity of this function it is essential for calculations in this and other functions like the size estimation described in the next section, even fundamental in the camera calibration (section 4.2.4).

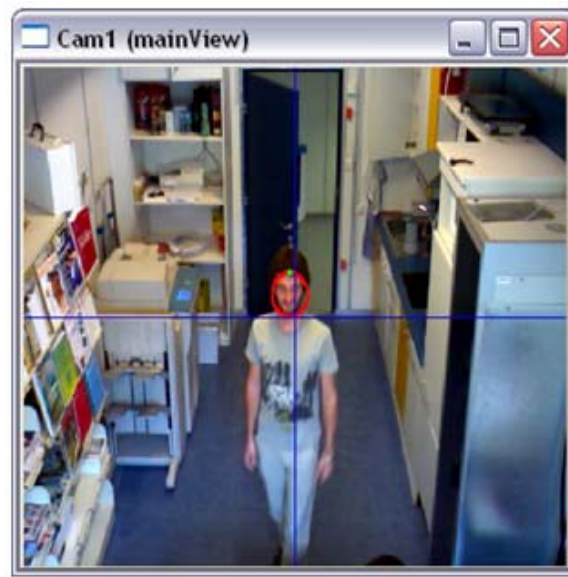


Figure 46: Detected face in CAM1 view field by means of *faceDet* module

detectAndDraw – The *detectAndDraw* function tries to find in the frame provided by the function *faceDet* all the possible faces and subsequently draw a circle around each face detected. The method used by the *facedetect* sample from the OpenCv package to detect faces was described in 3.3.1, some patterns are searched in the image that are considered by the classifier as part of a face. This is how the program detects the faces in the image.

Those patterns are sometimes fulfilled by other unwanted objects which can be considered as the designated *fake faces*. Figure 47 a) shows one test of the face detection in which three of the mentioned fake faces appear. To try to control the appearance of fake faces the following consideration has been taken: Since the camera used for this application is placed fixed to the ceiling, the detected faces, or in other words, the circles which correspond to those faces cannot be bigger than a determined radius. A detected face whose circle has a radius bigger than 35 pixels will be considered as a fake and will be ignored. Furthermore considering the placement of the camera CAM1 and the layout of the SmaKi, are deduced some regions in the room where the appearance of faces should not take place. These regions are estimated experimentally as it is showed in Figure 47 b). In this figure are represented two regions, region A and B; region A is the one calculated for one 171cm person. Region B is the extension of region A, estimated for one 200cm person. Thereby all faces detected out of the adding of these two regions will be considered as fake faces and consequently ignored by the system. With this method, the circle most on the right depicted in Figure 47 a) would be directly ignored.

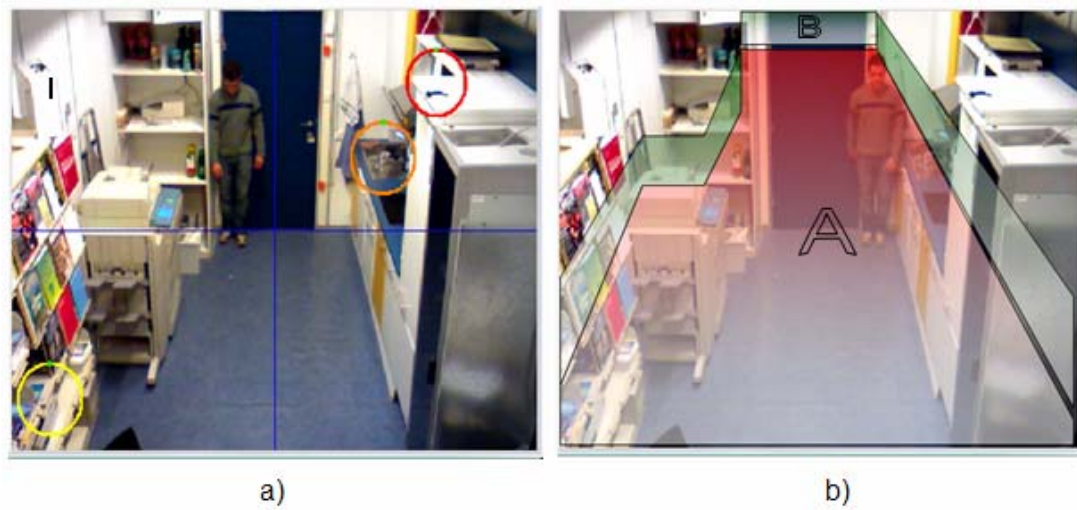


Figure 47: CAM1 view a) Fake faces in face detection test. b) Estimated regions to control the appearance of fake faces.

If the program detects a “true” face, `detectAndDraw` executes the function `center_axes` to get the centre point of the image (necessary for further calculations, like in module `sizeEst`, described in the next section) and returns the designated *top face* point (depicted in green in Figure 46), which corresponds to the highest point on the top of each circle drawn.

4.2.3 Size Estimator

It was discussed in section 3.1.3 the focusing of the system at hand on the “Child safety” application. The way this system faces this question is by means of this package which achieves the distinction adult/children, indispensable condition to continue doing any other consideration in this matter. The module `sizeEst` is the one entrusted to estimate the height of a detected person in the SmaKi. This module requires a great synchronization among several modules as well as good calibration of the camera for which this module is designated, CAM1. First of all when the system is started, the calibration for the CAM1 must be properly done by means of the procedure explained later in section 4.2.4. Furthermore it needs the *top face* point from the `faceDet` module (explained in the previous section) as well as the coordinates belonging to the centre of the image where the face was detected. The procedure to calculate the height of a desired person is described next.

With the same idea used for the camera calibration, it is possible to determine the size of any dynamic object in the room if we know the other two variables of the SPD formula (see section 3.3.4), the pixels it fills in the image and the distance between the camera and the object in matter. To determine the number of pixels corresponding to the size of an object, are necessary two points that delimit the object in height (one in the top, the other at the bottom

of the object), hence the number of pixels that comprise the object is clearly defined. But in this case the “size estimator” has only one, the one corresponding to the top of the circle of the detected face, the top-face point. Another problem when using this procedure, but at the same time the reason for having chosen face detection instead of body detection, is that there are some regions in the room where the whole shape from the person does not fit in the camera vision field. Due to the placement and focusing of the CAM1 these regions are those close to the camera (blue shading in Figure 49. CAM1 is placed in position A).

The solution to these two problems is directly given with the information provided by the floor sensors and the implementation of some geometric functions. Thanks to the floor sensors and the values resulting from the camera calibration, it is possible to know the exact position of the person and the distance dividing person and camera. Figure 48 (as well as Figure 49) is part of the calibration helper explained in section 4.2.4. In this figure, as a 3D view sketch of the SmaKi, the points of interest for the person height estimation are shown (the way this points are situated in the graph, considers the most general layout of the different factors involved in the size estimation).

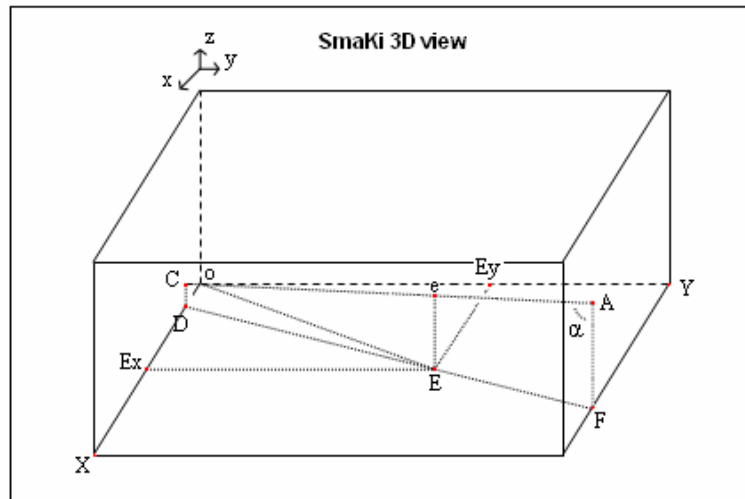


Figure 48: SmaKi sketch in 3D view with points of interest for the height estimation.

Following Figure 48, the camera is placed in A at an AF height of the floor, and focusing on point C. The position provided by the floor sensors when a person was detected corresponds to point E (given in to coordinates Ex and Ey), hence this is the only dynamic point to take into account. Some of the points (or distances) represented in the graph are outstanding for the camera calibration, these are described in detail in section 4.2.4.

Hence, with all the information described above, the calculation of the size of the person is possible, since the total height of the person is divided in to two sub-heights: H_p (Primary Height) and H_s (Secondary Height). Using different procedures H_p and H_s are calculated and with the addition of these two heights the real height of the person results, designated as H .

The primary height is the one comprised between the floor and the axis projection plane at middle image (depicted in Figure 49 as the straight line resulting of the union AC). The secondary height is the distance comprised between the middle image horizontal axis and the upper part of the head of the detected person. The way to calculate these two heights follows different procedures. H_p is the result of a simply geometric estimation by means of the known distances:

$$H_p = AF - \frac{\sqrt{(OY - OE_y)^2 + (YF - OE_x)^2}}{\tan(\alpha)}$$

H_s is calculated by means of the *SPD formula*. In the case in matter the searched value is the height, in other words, the unknown size (S) in the *SPD formula*. Remaining are, the number of pixels (P) and the distance (D) between camera and object (a person in this case). The number of pixels is the addition of the pixels existing between the axis projection plane at middle image and the upper part of the person's head, which is defined by means of the *top-face* point (provided by the `detectAnddraw` function). The distance between the camera and the person is designated as Ae (Figure 48) belongs to the first term in the equation. The resulting equation is the following:

$$H_s = \frac{\sqrt{(OY - OE_y)^2 + (YF - OE_x)^2}}{\tan(\alpha) \cdot \sin(\alpha)} \cdot S \cdot \text{Cnt}_{SPD}$$

Once these two heights are calculated, the total height (H) of the person is the result of the direct addition of these two variables. With the procedure followed to estimate the total height, three different cases can occur during the height estimation (with respective graphical representation in Figure 49):

- *case 1*: The entire shape from the person in matter fits inside the camera field view, consequently it is very probable that the axis projection plane at middle image (represented with the AC straight line) is placed under the *top-face* point where this plane cuts the detected person. In this circumstance H is the addition of the absolute values of the two sub-heights H_p and H_s .
- *case 2*: This case occurs when the axis projection plane at middle image cuts with the *face-top* point, or the upper part of the head. In this case H_s is cancelled and the resulting H has the same value as H_p . This second case normally occurs in proximities to the camera.
- *case 3*: In this case the entire shape from the person never fits inside the camera field. The *top-face* point is in this case under the axis projection plane at middle image. In this case H_p is greater than the real height, and H_s results a negative value. The addition of these two variables gives the right result H .

The formulas given for Hp and Hs consider the possibility of these three cases. Actually the resulting formula does not care the case in which the height is estimated, since they are not more than particular cases for the same formula.

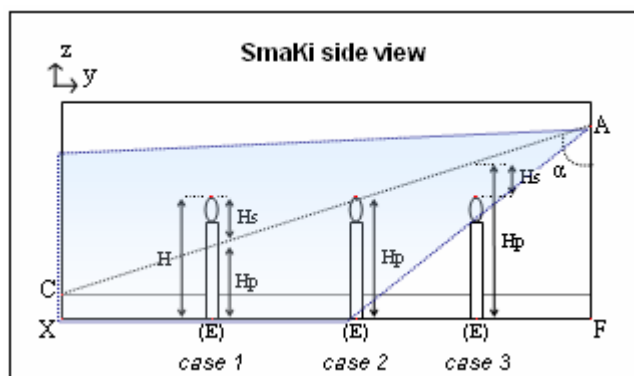


Figure 49: CAM1 field view and heights relation in SmaKi side view.

In addition, for the end-calculation of the height is important to consider the camera radial distortion described in section 3.3.5 – the size of the object in the image varies with a lineal proportion to the variation of the distance camera-object. In principle this aberration affects to both directions of the camera geometry, vertical and horizontal axis. Taking into account the layout of the SmaKi (Figure 22) provided in section 2.2.1, it is possible to optimize the radial distortion for the horizontal axis if the camera is centred in the room, since the maximal horizontal view range for this axis is considerably reduced. Consequently the error resulting of this adverse effect is reduced too. Comparatively the radial distortion in the current configuration is essentially greater for the vertical axis, with a three to eight relation, which represent the width and the depth of the room respectively. For this case, it is very difficult or even impossible, to estimate some metrics without considering this aberration. Thereby, the resulting height described previously is finally calculated with the formula showed next, which was experimentally calculated:

$$H = Hrd + (-0,4228 \cdot R + 180)$$

Hrd is the height containing the radial distortion error; R is the distance camera-person and H the resulting height of the entire height estimation procedure. The graph depicted next Figure 50 represents a collection of samples obtained using the “size estimation” application with a 171cm person as target. In the first graph (Figure 50 top) are considered for each sample H_s , H_p and the adding of these two variables; represented all them according to the distance between the camera and the person. The resulting height depicted in this graph is the one which is affected by the radial distortion (Hrd). It is possible to observe that the resulting height composes an ascendant trendline with the increasing distance, although the real size of the person is obviously a constant. The graph depicted on the bottom represents the same set

of samples obtained for H_{rd} , nevertheless in this case the formula to correct the camera imperfection is applied, having as result the expected constant trendline around the 171cm.

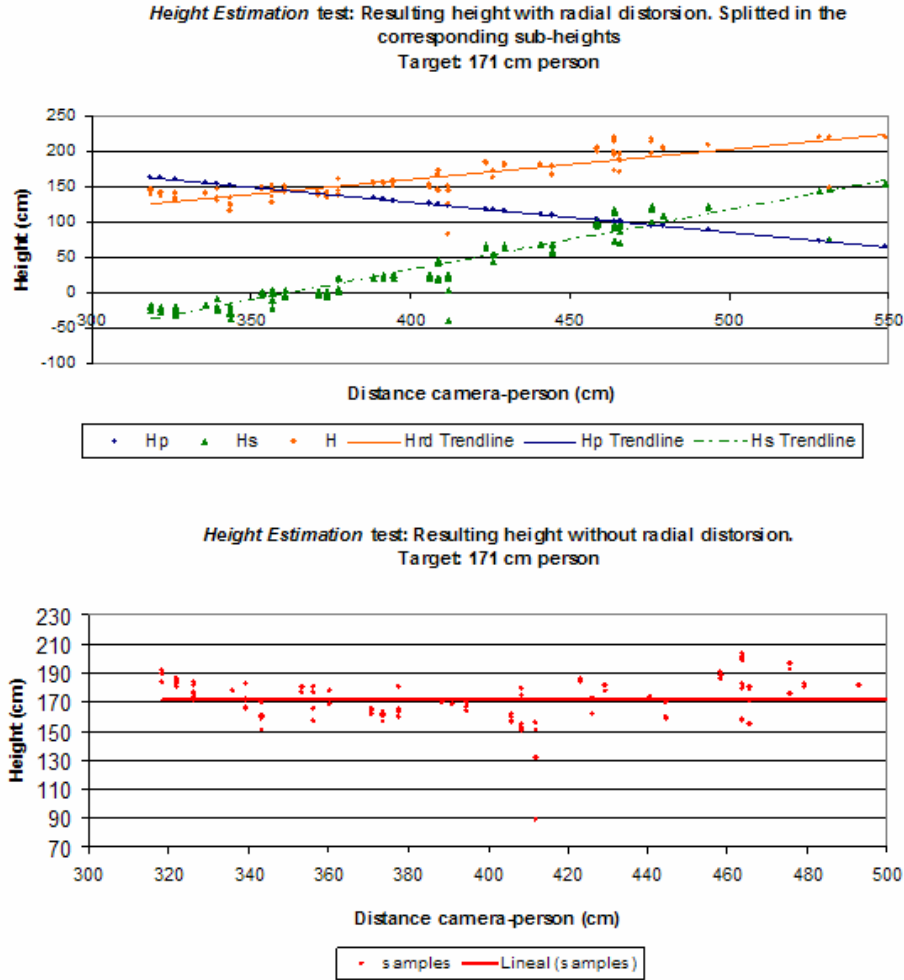


Figure 50: Graphics resulting of the height estimation test. (Top) Resulting height with radial distortion and composing sub-heights. (Bottom) Resulting height without radial distortion.

Furthermore it is possible to observe in the first graph the difference existing among the samples obtained of the H_s sub-height and the H_p one. The H_p samples keep a complete lineal series – the points that represent each sample fit absolutely to the trendline – and those from resulting from the H_s represent certain grade of dispersion (regarding to the trendline). The reason of this difference resides in the geometrical variation resulting from the face detect (remember the H_s is calculated with the top-face point). On the other hand, the H_p is calculated directly through some geometric estimations and the coordinates provided by the floor tactile sensors; the linearity is an intrinsic property of the equation that generates the sub-height H_p .

From all this procedure the importance of the correct detection of the faces is clearly deduced. Since this module uses information from the `faceDet` module it is possible to have dragged an error coming from a non controlled *fake face* (explained in section 4.2.2), which can turn into *fake heights*, heights that are impossible to correspond to any person in the room. It is possible to control the fake heights when they result impossible heights (for example 2,5 metres or 50cm are not very probable as person heights).

4.2.4 Camera Calibration

It was already mentioned in section 3.3.4 that the camera calibration in this work is not understood in the sense of creating a 3D vision, but in the sense of a procedure that facilitates the coordinates of the camera in the room, as well as a mechanism that achieves the estimation of metrics. In this section two possible ways that were developed to carry out this task are described, one the called *dynamic calibration* and the other *manual calibration*, both designated for CAM1 which has the main view in the SmaKi. Furthermore for the camera CAM2 another application, also defined as calibration, was developed, although no distances or sizes calculated are wanted to be calculated in this case. That calibration, as explained later, is a manual calibration too, which requires in this case the proper focusing of CAM2 in the bookshelf for the right operation of the functions this camera carries out.

CAM1 dynamic calibration – There is a function in this module which is entrusted to do the dynamic calibration of the camera CAM1, the function called `camCal`. The aim of this function is to calculate the current coordinates of the camera in the SmaKi. The procedure is based on the *SPD formula* principle (see section 3.3.4). When the `camCal` is executed the system calls the function `findDoor`, to have a known size element in the SmaKi as reference. After finding the shape of the door the application uses a modified Lucas-Kanade optical flow algorithm, as the one from the *lkdemo* sample explained in section 3.3.3, to track the door. To achieve this tracking the system sets two *LK-points* on the centred in the borders from the door shape, one at the top, the other at the bottom (green points in Figure 51).

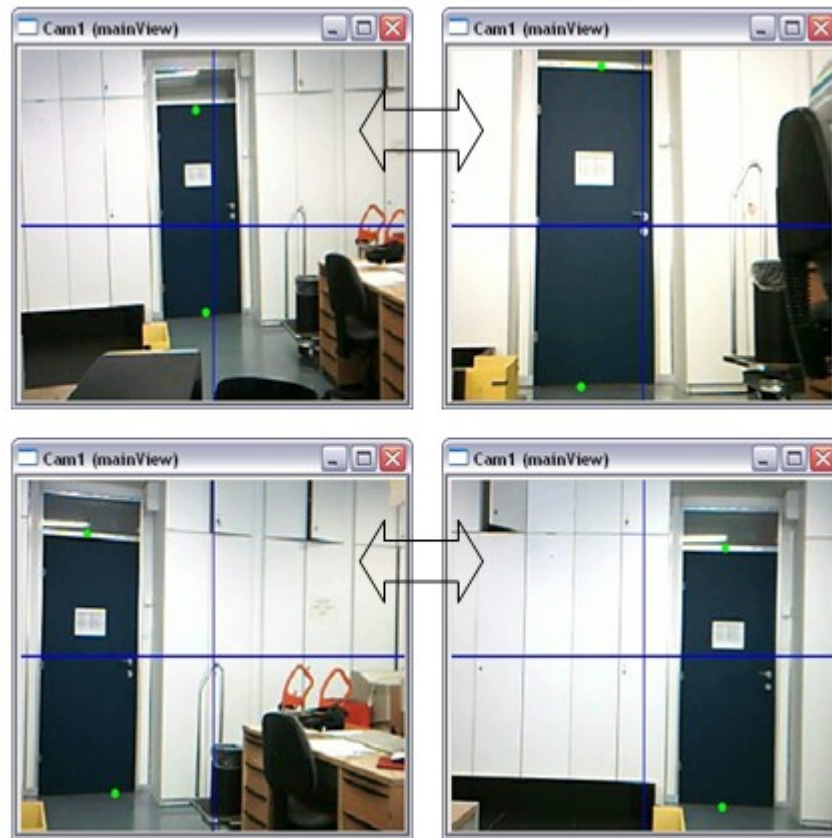


Figure 51: Door tracking in dynamic calibration

With this procedure, not only is the door tracked in all times, but also the pixels comprised between these two *LK-points* are easily calculated. Since the number of pixels from the door height in the image is a known value the *SPD formula* can be used to estimate the relative position of the camera in the room (remember that the real size of the door is a known value, an important fact to carry out the identification of the door by means of the *doorFinder*). Since the position from the door in the room is a known value too, the current position from the camera in the room is deduced with a simply coordinates change. This is called a dynamic calibration, since as explained above and depicted in Figure 51 (four different screenshots in the same test, where the camera has been moved in different directions and distances), the *LK-points* follow the door borders during the running of the application and this way, although the camera is displaced from its initial position, its location in the room is always a known value. Of course this application only works if the whole shape from the door is drawn in the image, otherwise any of the two points in play can disappear and with it the control of the door.

CAM1 manual calibration – Another module was developed to provide the possibility of a manual calibration of CAM1. This module includes a help program that can be launched at the beginning of the application. This program indicates the steps to follow in order to carry

out the proper calibration of the camera; where the measures must be done and how to configure the program. This module consists mainly of three components. The first one is the so called *calibration helper* which is an image stored in the program, the one depicted in Figure 52 and that shows the important points to consider in the camera calibration. The second main component of this module is a common text file where the different steps to follow in the calibration are described; by means of this indications the different measures in the room should be done, taking as reference the *calibration helper*.

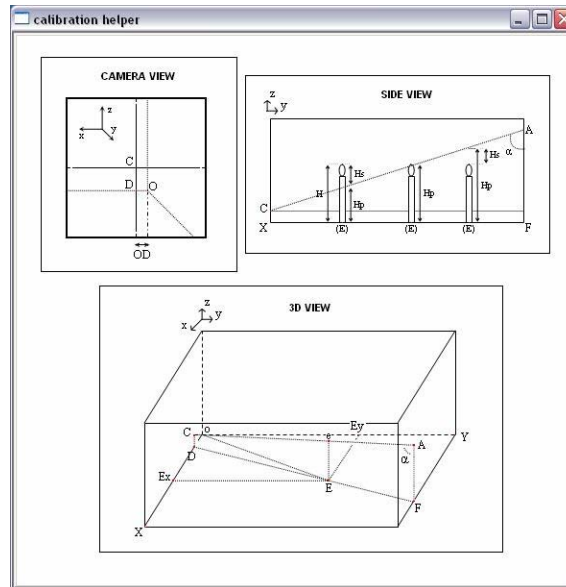


Figure 52: CAM1 manual calibration helper

After the measures are done, the resulting values must be stored in the same text file, since these are the values that the system needs to accomplish the calibration. This two components, picture and text file, are launched simultaneously when the manual calibration function is executed. The third component of the module is a function called `manCamCal` that reads the measure values previously stored in file and integrates them in the system for further use in other applications. If the system is started and no manual calibration is executed, the system loads the last values stored in the text file and use them for the camera calibration.

CAM2 calibration – In the same module `camCal` there is another function to calibrate CAM2 when the system starts, the function called `cam2CalLoop`. When this function is executed a template is shown (blue squares in Figure 53) within the camera view. To calibrate properly CAM2, the camera must be (manually) moved until magazines (actually their centres) and template fit in. This template is made by mean of function `mInArray` (explained in section 4.2.5) which contains in memory 11 coordinates experimentally calculated, one for each magazine centre.

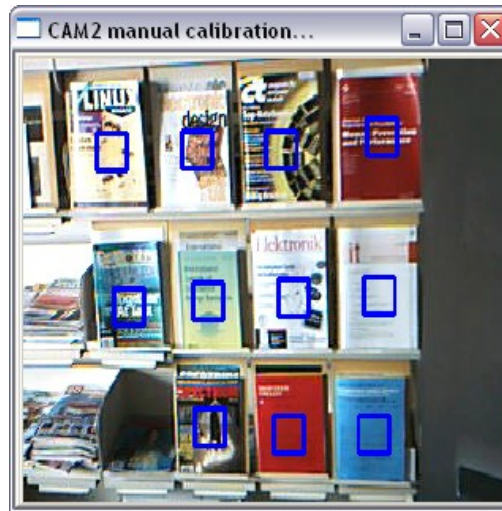


Figure 53: Bookshelf view in CAM2 manual calibration

4.2.5 Magazine Controller

In order to achieve the recognition of the scenario “Person takes magazine” (described in section 2.1.4) were developed some image processing functions that involve procedures like background subtraction and shape recognition (introduced in section 3.3). The control of the magazines at the bookshelf is carried out by the module `magaz`. This module is divided in four functions which are `mInArray`, `takePic`, `magaz` and `findMagaz` and are described as next:

mInArray – Since the camera in matter is fixed focusing to the bookshelf, the place that belongs to each magazine in the camera image is a known value. This function is the one which gives the position corresponding to each magazine in the image, since those positions were previously experimentally measured. These positions are each given as a point in two coordinates, which belongs to the centre of each magazine. From the 11 magazines this position is stored in an array for subsequent use from other functions.

takePic – This is the function used in this module to capture and store the pictures for its subsequent processing. First of all the routine will check if the image used as background is already stored, if not, a picture from the bookshelf is taken and stored at the same time. If the picture corresponding to the background already exists, the function `takePic` will make another capture for future analysis.



Figure 54: Bookshelf image capture. a) Bookshelf background, b) Bookshelf current state.

magaz – By means of the function `magaz`, it is possible to know if any magazine was removed from its original place. When the function is called, the background image from the bookshelf and the one to compare, both captured by function `takePic`, are loaded and subsequently compared. The comparison is made by mean of a subtraction function, if the resulting image is not an entire black one (which is the case of Figure 55), a magazine will be considered as taken. This resulting subtraction, gives the magazine in negative colour since the gap left by the magazine is white.



Figure 55: First subtraction image resulting. Image in negative.

To convert the magazine to its real colour, a new subtraction is done, in this case, between an entire white image and the one from the previous subtraction. The resulting image is the magazine missing in true colour, isolated from the rest of the bookshelf. This image is subsequently stored for analysis carried out by the next function explained, `findMagaz`.

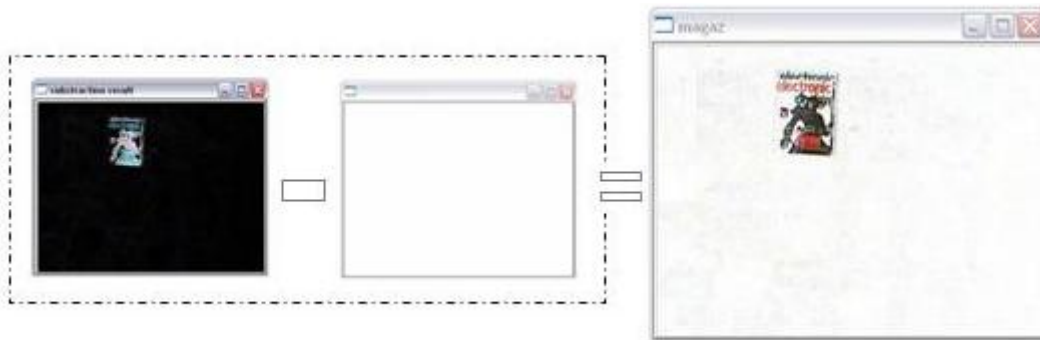


Figure 56: Second subtraction image resulting. Image in true colour

Furthermore, to control what person takes the magazine, the function `magaz` is connected with the module `proofPerson` (see section 4.1.5), a wider description about this relation is given in section 4.4.3.

findMagaz – This function is essentially a shape detector. The routine `findMagaz` uses the image resulting from the `magaz` function, and tries to find rectangles by means of different image analysis methods and geometric estimations. This is how the system detects the missing magazines, moreover with the pattern resulting from the function `mInArray`, the function `findMagaz` recognizes the magazine detected.

The routine developed for detecting a magazine as a square, is very similar to the one used in section 4.2.1. First of all an empty sequence of squares that will contain points, four points per square, is created. A filtering of the noise in the image is done with a down-scale and a subsequently up-scale from the image. But the key for the optimal detection of the squares, in this case, is the threshold used. Several threshold levels are tried on the image, concrete levels comprised in a selected margin specially optimized by mean of experimental tests. Since the contours described by the magazines are not perfectly defined, the routine entrusted to find the squares, identifies several squares for each magazine. To avoid this result, the program resorts to an iterative process in which the duplicate squares are ignored. At the end of the detection, in the sequence of squares only one square per magazine remains. Furthermore, although it results obvious, it is very important to consider that the magazines in matter are no intermixed or partially occluded or the mentioned match will no longer be realized [Sin02]. Therefore the function `nearObj` explained in section 4.1.5, provides the necessary tool to avoid this unwanted situation. A greater explanation of the relation between these two functions is given with the description of the scenario “Person takes magazine”. In section 4.4.3. Figure 57 shows the result of the magazine detection in two different cases in which the program has drawn green squares where the contours were identified.

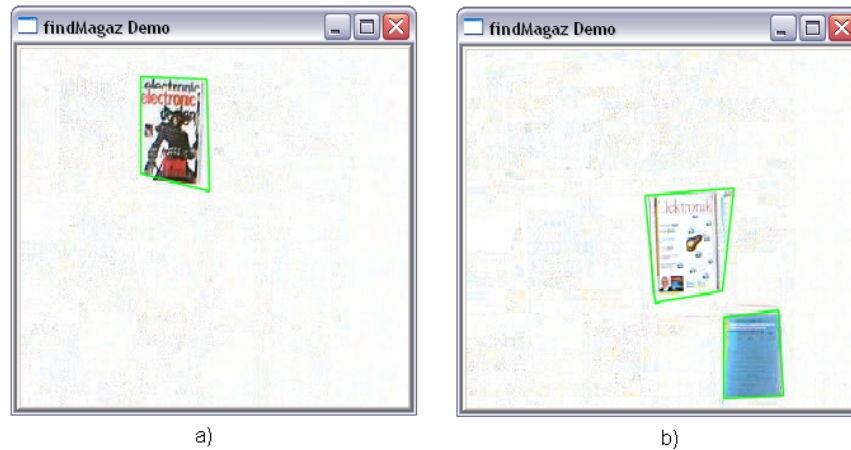


Figure 57: Magazine detected as square: a) One magazine case. b) Several magazines case.

After the different squares (magazines) in the image are detected, the system compares the centre of each square found with the positions given by the `mInArray`. If those centres match inside a margin of pixels with regard to the reference pattern established by the function `mInArray`, the missing magazines will be detected. The results remain stored and the state of the bookshelf is always controlled, what magazines are missing and which are not.

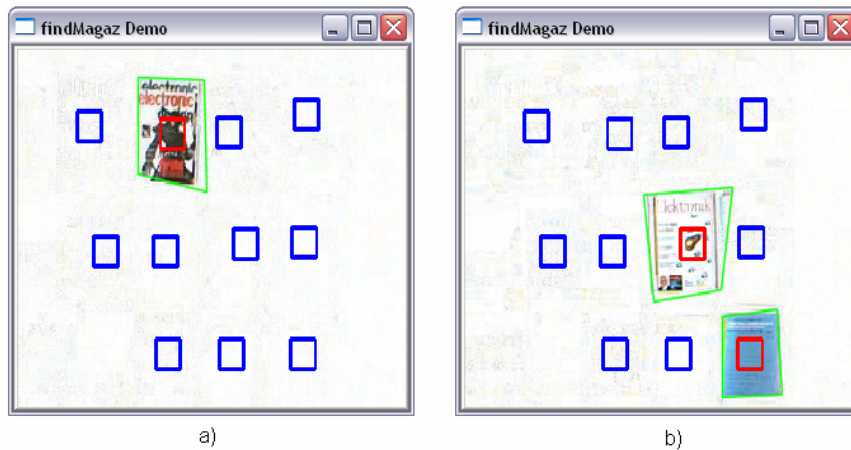


Figure 58: Missing magazines recognized. a) Magazine no. 2 recognized.
b) Magazines no. 7 and no. 11 recognized.

4.2.6 Cup Detection

In order to provide support to the recognition of those scenarios in which the coffee machine takes part like “Person makes coffee” or “Child makes coffee” is developed a software module in which the ARS-PC Vision recognizes when a cup appears in the camera view

field. The detection of the coffee cup is carried out by the module `cupDet` which contains de only routine called `cup`. This function is basically a shape recognizer, in this case circles. By means of some processing image functions, the captured image is prepared for an optimal detection of circles, which is done through the function `cvHoughCircles`. The functions used to treat the image are among others, the smoothing and dilation techniques explained in section 3.3.6. The function `cvHoughCircles` finds circles in greyscale images using Hough transform. These functions can be configured to determine some properties from the circles to be detected, like minimum radius, maximal radius, or the minimum distance among circles (among its centres).

Sometimes, the program detects the denominated *fake cups* (unwanted circles) due to changes of light or simply because of bad detection, among other reasons. With the right configuration of the function `cvHoughCircles` the appearance of the fake cups can be reduced. Since the camera in matter is fixed at any distance of the coffee machine, the resulting radius from the detected cups must be comprised between two values to consider them real cups. Circles with a radius smaller than the minimum selected or with radius bigger than the maximal one are considered as fake cups and consequently ignored.

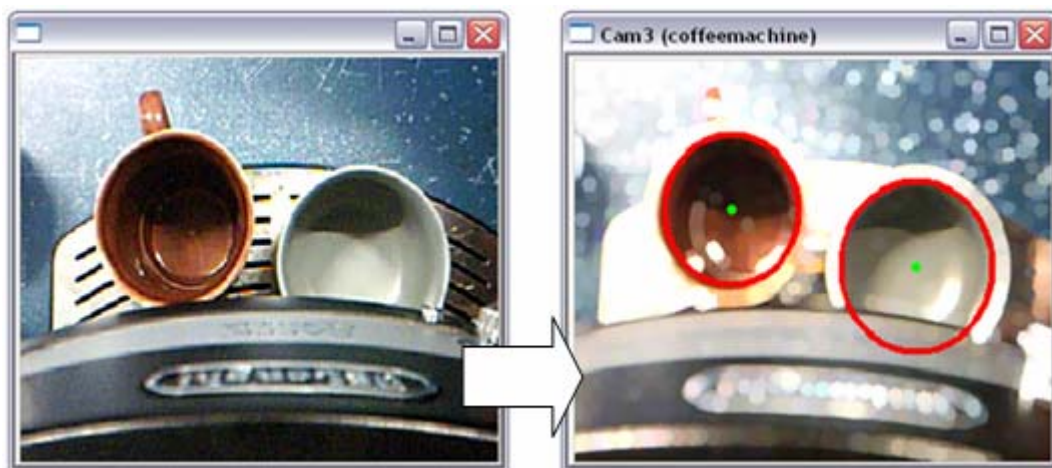


Figure 59: CAM3 view field . (Left) Current image without image processing, (right) image processed and analyzed, cups detected.

With the other parameter mentioned, which is the minimum distance among circles, and considering the minimum radius mentioned before, two circles will be considered as fake cups if the distance among them is less than the resulting of the double of this minimum radius. The third and last method applied to try to control the appearance of fake cups, is to restrict the number of detected circles in the image at the same time; only to coffees can be prepared at the same time, which that means two cups.

Figure 59 (right) shows the result of the “cup detection” in which two coffee cups were detected. The image was treated with functions dilate and smooth [ORM01,JH00] and the founded contours are drawn with red circles by the same function `cup`.

4.3 Interface

This section as well as the corresponding package with the same name, is designated this way since it has the same purpose as the strictest meaning of the word in matter. The Encyclopædia Britannica defines interface as “*the place at which independent and often unrelated systems meet and act on or communicate with each other*” [BOE07]. Consequently, in this section it is described, firstly the way the cameras capture the images and how these images are subsequently visualized by means of the module `cameraLoop` (section 4.3.1) and after that in section 4.3.2, the developed procedure to control the system operation (programs that will be executed or not), as well as the “running modes” available (sections 4.3.3, 4.3.4 and 4.3.5)

4.3.1 Camera Loop

The `cameraLoop` module is the one which manages the image capturing and the consequently video reproducing. This is made by means of two functions mainly, the `cameraLoop` (one specific for each camera) and the `checkTimeForCam` function.

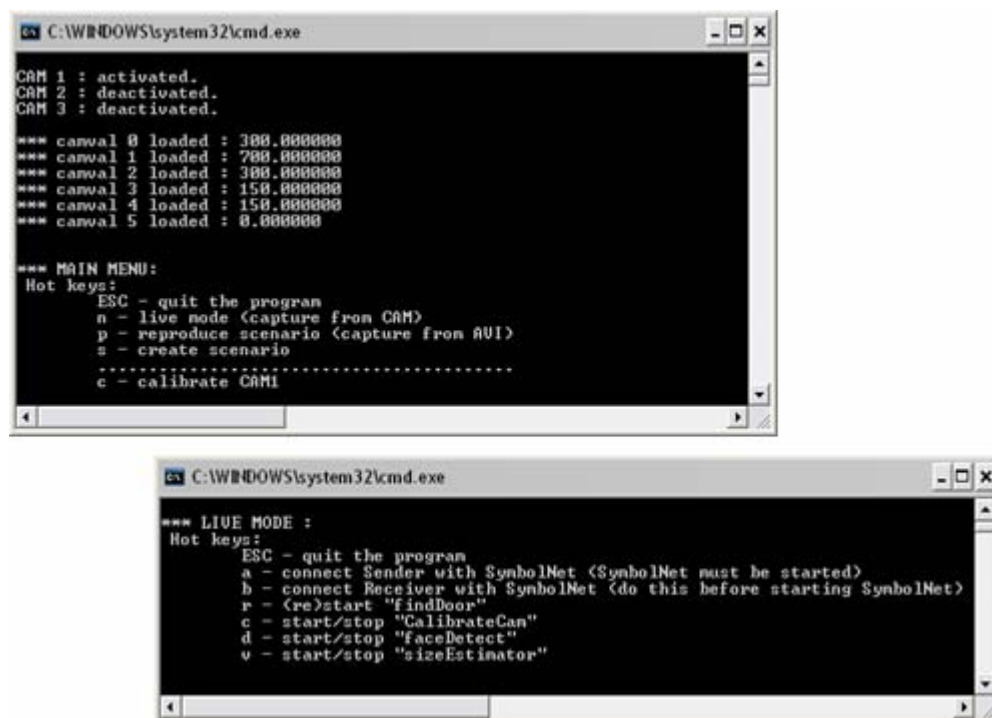
cameraLoop – This function has three different counterparts, one for each camera considered in this work. The aim of a `cameraLoop` function is, at first capturing the frame, check if the camera in matter provides the frame properly and shows the received image in the corresponding window as a video sequence. There are three different `cameraLoop`, hence there are three visualizing windows (“main view”, “bookshelf” and “coffee machine”) one for each camera. This routine controls the time the program remains looping since this is an important fact for the right visualizing of the image. Furthermore, if the system runs in the *Recording Mode* (explained later in section 4.3.5) the routine writes the captured frame in a specific video file in order to record the corresponding video sequence.

checkTimeForCam – This function is called only in the *Database Mode* (running mode explained in section 4.3.4), this is when the image capturing is carried out by means of a previously stored video file. The `checkTimeForCam` function, is in this case the one which controls the three `cameraLoop` functions by executing them in a sequential way so that each video is reproduced with the proper frame rate.

4.3.2 Function Management

The function management, in other words, the way the system controls what functions or configurations – like the operation modes described later in the next section – must be taken into account, is carried out by means of two modules, the `keysControl` and the `checkAndRun`.

keysControl – This module contains several functions that are entrusted to display at all time the possible configurations the system admits and the corresponding keys that activate (or deactivate) the respective functions. In Figure 60 (top) is shown the display in the command window of some configuration values, like the cameras that are currently activated as well as the values resulting from the camera (manual) calibration. Moreover the “Main Menu” is shown together with the corresponding keys that activate the possible program options. This menu will be displayed in a different way depending on what cameras are activated. In case the camera CAM2 is activated, a new option appears in the menu to facilitate the calibration of the camera in matter. By choosing one of the options available, a new menu is displayed Figure 60 (bottom), in this case the one corresponding to the *Live Mode* menu.



The figure consists of two screenshots of a Windows command prompt window. The top screenshot shows the 'MAIN MENU' with options to quit, enter live mode, reproduce a scenario, create a scenario, or calibrate a camera. The bottom screenshot shows the 'LIVE MODE' menu with options to connect to SymbolNet, start/stop various functions like 'FindDoor', 'CalibrateCan', 'faceDetect', and 'sizeEstimator'.

```

C:\WINDOWS\system32\cmd.exe

CAM 1 : activated.
CAM 2 : deactivated.
CAM 3 : deactivated.

*** canval 0 loaded : 300.000000
*** canval 1 loaded : 700.000000
*** canval 2 loaded : 300.000000
*** canval 3 loaded : 150.000000
*** canval 4 loaded : 150.000000
*** canval 5 loaded : 0.000000

*** MAIN MENU:
Hot keys:
ESC - quit the program
n - live mode <capture from CAM>
p - reproduce scenario <capture from AVI>
s - create scenario
.....
c - calibrate CAM1

*** LIVE MODE :
Hot keys:
ESC - quit the program
a - connect Sender with SymbolNet <SymbolNet must be started>
b - connect Receiver with SymbolNet <do this before starting SymbolNet>
r - <re>start "FindDoor"
c - start/stop "CalibrateCan"
d - start/stop "faceDetect"
v - start/stop "sizeEstimator"

```

Figure 60: System options management by means of `keysControl`. (Top) Main menu, (bottom) *Live Mode* menu.

checkAndRun – This module is composed of only one function designated with the same name. This routine is the one entrusted to go through all the different functions belonging to each camera and execute those which are activated (by means of the `keysControl` or as default depending on the running mode). Furthermore the function `checkAndRun` sets the proper configuration for each video writer (explained later in section 4.3.5) in case the operation mode is the one to record scenarios.

4.3.3 Live Mode

The *Live Mode* is the one developed to work synchronized with the *ARS_Live* (see section 3.2), that means the system will work processing real-time information. The ARS-PC Vision receives on the one hand all the symbols incoming from the ARS-PC and on the other hand the information generated by the cameras.

Figure 61 represents the main loop the system follows in the *Live Mode* configuration. First of all as an indispensable condition for the proper system operation the program waits until the connection with the ARS-PC is established. Once the connection is established the program remains alternating mainly among three modules, the `receiverLoop`, the `checkAndRun` and the `cameraLoop`. Only if any “valid” detection is made by any of the functions called by the module `checkAndRun`, the corresponding `NewSymbol-Message` will be generated and subsequently sent to the ARS-PC Vision by means of function `sendSymbol`, for after that continuing alternating among the functions already mentioned. Due to the procedure used in this implementation is mandatory that none of the mentioned functions block the course of the system. Therefore functions like the `receiverLoop` or the `cameraLoop` are controlled by time (and other specific conditions that consider special situations).

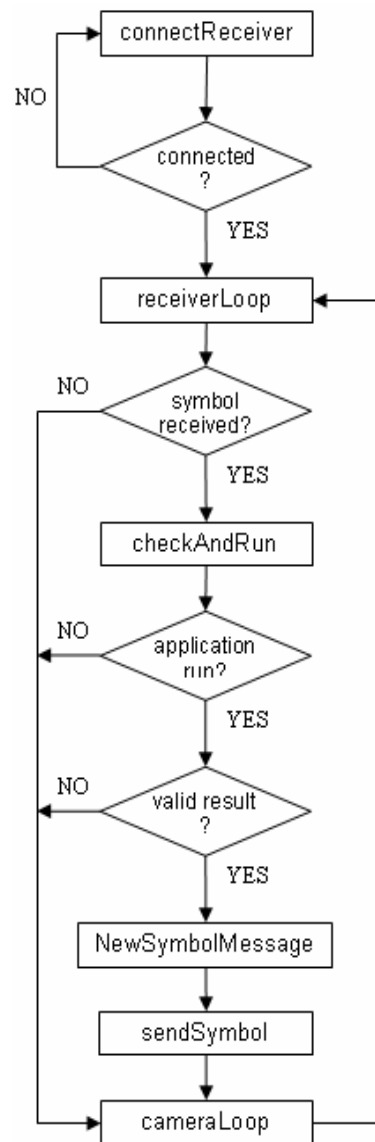


Figure 61: Main loop comprising the *Live Mode*

4.3.4 Database Mode

Since all the information which does not belong to the one provided by the cameras, is received in the ARS-PC Vision through the ARS-PC module, the first one does not differentiate if the incoming symbols are generated by current sensor values (*Live Mode* case) or, as in case of *ARS_Database* (referenced in section 3.2), from values previously stored in the Sensor Database. In this sense, this operation mode and the one described in the previous section, work the same way. The only question that differentiates these two running modes, the one that really cares in this case, is that the video capturing is made from previously

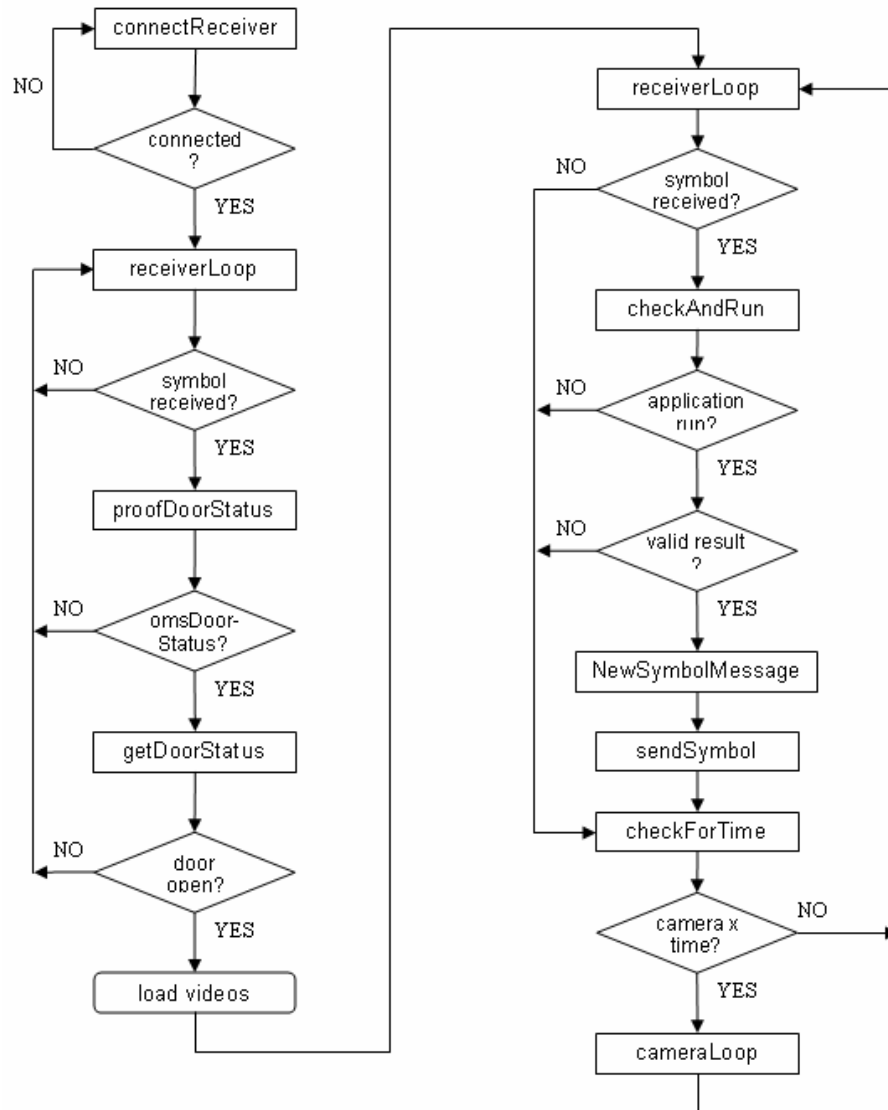


Figure 62: Main loop comprising *Database Mode*

stored video files and hence the different system functions will be executed over those files. Due to this fact, ARS-PC and ARS-PC Vision must be well in-time synchronized. That means that the ARS-PC Vision must start to reproduce the corresponding video files at the same time the scenario does and remaining synchronized during the reproduction of that scenario. It would be senseless to analyze a video sequence where a person is close to the table and the received sensor information belongs to the door contact switch, activated some seconds later or vice versa. To achieve this synchronization the program loads the corresponding videos when the microsymbol “Door status” is received, because the main door in the SmaKi was opened. This is actually the instant in which the recorded scenario starts.

Furthermore, the system must know which video correspond to each scenario. The program will load the videos corresponding to each cam, when their names (the file name) match with the *timestamp* from the received “Door status” symbol (symbol structure explained in section 2.2.4). The videos will be loaded when the first “Door status” symbol is received from the ARS-PC as they were recorded when the same symbol occurred during the recording of the scenario in matter (task entrusted to the *Recording Mode*, explained in the next section). The reproducing of the three videos and the appropriated time synchronization is controlled with the function `checkTimeForCam` (section 4.3.1). In Figure 62 is depicted the procedure followed through the *Database Mode*, the necessary steps until the videos are loaded, to remain after that in the same loop as the *Live Mode* with the only difference of the video synchronization.

4.3.5 Recording Mode

In section 3.2 was discussed the necessity of a third operation mode in order to record scenarios as part of the test purposes concerning the entire ARS-PC system. Through the recording mode, no analysis of the image is done, only a visualization of the camera capture and the corresponding video recording is carried out through this procedure. In Figure 63 it is described, using the different modules composing the system, the run procedure followed in the recording mode. First of all, the system waits until the connection with the ARS-PC is established, after that it remains inside the `receiverLoop` until the right symbol “Door Status” reaches the ARS-PC Vision to subsequently create the video files with the whole procedure previously explained. When the video files are created, the system remains alternating between the `cameraLoop` (where the video sequences are generated and stored) and the `receiverLoop` until the scenario finishes.

For each camera a different video file is created and for each file a video file writer must be assigned. A video file writer needs to be properly configured for the right recording of the video. The different information necessary for its configuration is: the name of the file in which the video will be stored, the frames per second rate (fps), the size of the video frames and the name of the codec in FourCC (Four Characters Code) [MDN07].

In order to facilitate the further access to the created video files, the name of each video file is automatically designated with the timestamp of the received microsymbol “Door Status”, when the system detects that the main door is opened. For each file the name is designated as follows “<timestamp>_CAM<camera number>.avi”. The timestamp is the one from the symbol “Door status” and the “camera number” is an integer that refers to the camera which is assigned to take that video; numbers 1, 2 or 3.

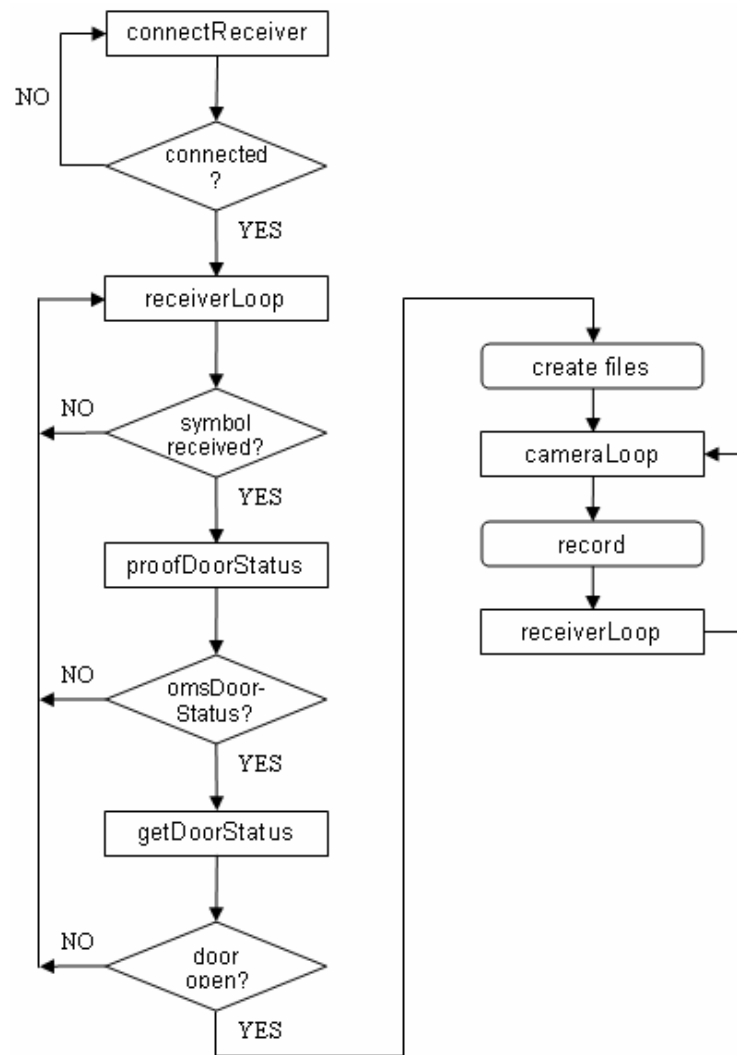


Figure 63: Main loop comprising the *Recording Mode*

Which codecs are supported depends on the back end library. On Windows HighGUI (see section 2.3.2) uses Video for Windows (VFW), on Linux ffmpeg and on Mac OS X the back end is QuickTime [OCL07]. Since this work is developed on Windows, the possible video codecs to choose must be in VFW format. The one used to configure the three video file writers is MPEG-4, since it is a common codec and its relation compression-quality acceptable enough (in FourCC MP42 [MDN07]).

The fps selected should be under the maximum fps indicated in the specifications (see section 2.3.3) of each camera. The fps used for each video file writer were chosen considering the requirements of the applications each video is designated for. For example, the camera CAM3 is focused to the coffee machine, in this case it is senseless to choose a high fps, since for the detection of the cup, temporary information is not necessary. The camera CAM1, which has a bigger line of sight, has to have more frames per second since the applications

designated for this camera need to be in time synchronized and the time precision is necessary. The frame size field in each video file writer is selected for each camera according to the maximal video resolution each camera can give.

With all these configurations the system is prepared to record the desired scenario when the “recording mode” is executed. At the end of the sequence, there will be so many videos as cameras in play (with a maximum of three cameras) with the proper configuration for their reproducing within the Database Mode.

4.4 Applications

Since the system is designed for the final applications mentioned before (see section 3.1), the ARS-PC Vision is optimized for applications “Scenario Recognition” and “Child Safety”. In this section the connections between the different modules previously explained are described to achieve the mentioned applications.

4.4.1 Adult/Child Distinction

Actually this section does not describe the way followed to recognize a concrete scenario, but it comprises the base to those scenarios where the “Child safety” represents the final purpose. The distinction between an adult and a child consists basically in the detection of a person and its subsequent height estimation.

The procedure followed to achieve the distinction among adults and children involves several modules or functions, starting with the activation from the tactile floor sensors, going through the ARS-PC, then through the ARS-PC Vision and finally returning again to the ARS-PC for the consequent support in the recognition of the scenario. In Figure 64 the different steps and the necessary modules to accomplish the height detection are enumerated.

For the estimation of the height of one person, the only information the ARS-PC Vision requires from the ARS-PC (remember that all the information provided by the sensors arrive at the ARS-PC Vision through the ARS-PC), is the current position of that person in the SmaKi. As shown in Figure 20 the first symbol generated through the tactile sensors is the microsymbol Object. Then, only when this symbol is received in the ARS-PC Vision, the adult-child distinguish mechanism will start. With the activation of a tactile floor sensor many processes are unleashed until the ARS-PC recognizes that activation as a microsymbol object and subsequently sent to the ARS-PC Vision via TCP. Since, as explained in section 3.4.1, the ARS-PC sends every symbol generated to the ARS-PC Vision, it is necessary to check what kind of message has been received, because only one gives the right information to accomplish the target of the application. Only if the message received is a microsymbol object with property Position and scheme Rectangle (verification carried out by function

proofRectangle) the function `getPosObj` will try to get the value of the corresponding property Position within the symbol object.

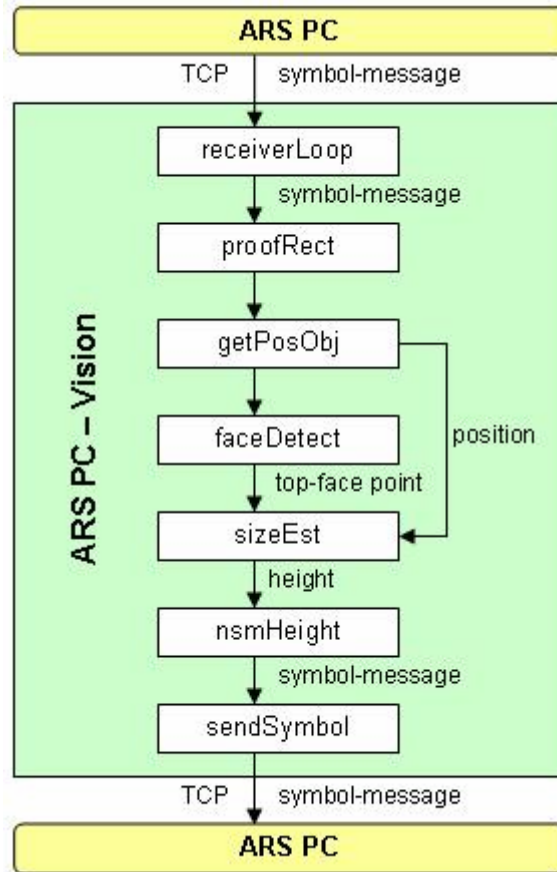


Figure 64: Unleashed procedure in a successful *Adult/Child Distinction*.

Once the system has the current position from the object, executes the face detection (function `faceDetect`) looking for a possible human face in the image. If the floor sensors were activated by a person, all the involved processes worked properly, and the circumstances for the image processing were propitious, the face of that person will be detected in the image and the next routine from the application will be executed. The next task in the application is to calculate the height of the person detected, which is made by mean of function `sizeEst`. For this calculation this method needs the top-face point given by the `faceDetect` function, the position of the person in the SmaKi and some values resulting from the camera calibration. With all this, several geometric calculations and some mathematical estimations, the height is deduced. Of course the detection will be only carried out in the regions where tactile floor sensors are placed and at the same time regions covered by the CAM1 view field (this region is clearly defined in the layout of the SmaKi given in

Figure 22, where the camera view field starts approximately at the point marked with an “x” and ends in the left extreme of the room).

When the desired value is calculated, it is sent to the function `nsmHeight` and it is encapsulated there with the proper symbolic structure that makes it able to return to the ARS-PC via TCP (by mean of function `sendSymbol`) to continue the processing by other modules. Different modules in the ARS-PC use this information provided by the ARS-PC Vision to generate scenarios where a child is involved, like the “Child makes coffee” scenario or the “Child near stove” one.

4.4.2 Scenario: Adult/Child makes coffee

It was already mentioned in section 2.1.4 the existence of a scenario-symbol “Person makes coffee” and another one designed as “Child makes coffee” in order to consider the hazardous situation, as part of the application “Child safety” (discussed within the system design in section 3.1), in which a child tries to manipulate the coffee machine. For the generation of the scenario “Adult makes coffee” or “Child makes coffee” two facts must be considered. First of all the preparation of the coffee must be recognized and then if the person in matter is detected as an adult or as a child. The first question is solved with the vibration sensor installed in the coffee machine (section 2.2.2), and the second one is carried out by means of the process explained in the previous section, the Adult/Child distinction. In addition, the ARS-PC Vision gives support to the vibration sensor, since with the succession of the processes shown in Figure 65 the system informs the ARS-PC when a cup is detected in the camera view field.

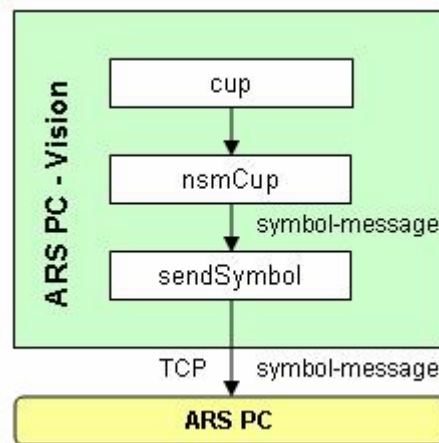


Figure 65: Unleashed procedure in a successful *Cup Detection*.

The function `cup`, which is assigned to the camera CAM3, checks if a cup appears in its vision field by mean of a circles detector. In case the detection is satisfactorily made, a NSM is created by the `nsmCup` routine, and sent by means of the function `sendSymbol` to the

ARS-PC. The ARS-PC manages this information contrasting it with the one from the vibration sensor at the coffee machine on the whole from the application from the previous section and finally generates the scenario in matter.

4.4.3 Scenario: Person takes magazine

Even though the previous scenario did already exist in the ARS-PC, the ARS-PC Vision provides a great support in case of sensor break down or simply for a greater contrast in matter of detection. The scenario “Person takes magazine” described in this section represents a new scenario for the ARS-PC (already referenced in section 2.1.4). Due to its detection characteristics and the sensors in the SmaKi previously installed, a scenario with those characteristics was impossible to be recognized until the assembly of the cameras in there.

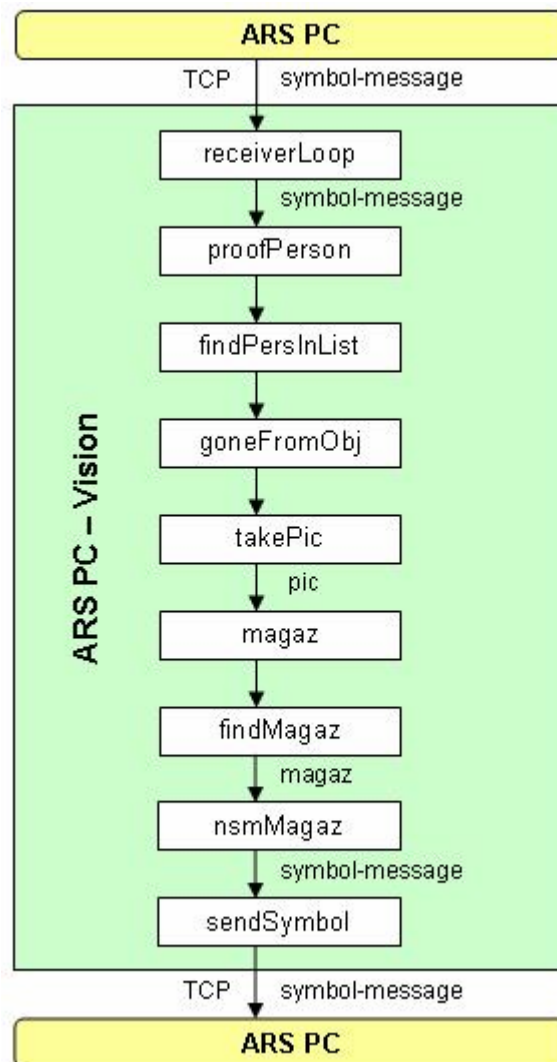


Figure 66: Unleashed procedure in a successful detection during the *Magazines Control*.

The generation of the scenario “Person takes magazine” is the consequence of the control from the magazines on the bookshelf and it is generated when the system detects that a person takes a magazine from the bookshelf away, as well as updated when the magazine is set back in the bookshelf. The procedure the system follows to detect this scenario is shown in Figure 66 and described next:

At the beginning of the program, after the TCP connection between the ARS-PC and the ARS-PC Vision is made and when the first symbol message arrives at ARS-PC Vision, the function `takePic` makes a capture from the current image of the bookshelf. This picture is stored as bookshelf background for future comparisons. After that, the system remains in the main loop (represented in Figure 61 for the “live” detection), and does not react until a symbol `Person` is received from the ARS-PC. If a symbol `Person` is received (message proofed by function `proofPerson` inside the `receiverLoop`) it is because a person is recognized at that moment in the `SmaKi`. After that, the ARS-PC Vision makes a register of all persons in the room (with functions `proofPerson` and others from module `list`). By means of function `nearObj` the system checks for each incoming UPM if any of the registered persons is near the bookshelf. If this is the case, the system waits until that person moves away from the bookshelf proximities to make a new capture (function `takePic`). This way the system makes sure to take a clean picture of the bookshelf (without occlusions). After this second capture is done, both captures, the first considered as background and the last one, are compared in order to check if changes were made in the bookshelf (function `magaz`). If this is the case, the system tries to find the missing magazine in the image (with `findMagaz`).

With this procedure the system keeps the current status of the bookshelf, what magazines are missing and which not. When a change in the status of the bookshelf is detected, the corresponding NSM is generated (in this case with the `nsmMagaz` function) with the current status of the bookshelf and sent via TCP back to the ARS-PC (by means of function `sendSymbol`). The symbol is sent as a sensor-value symbol. That means that the ARS-PC must generate, after receiving the message from the ARS-PC vision, the corresponding symbols for each level until reaching the scenario-level.

In Figure 67 it is shown the graphic visualization of the `SmaKi` (developed in [Sch07] as part of a previous thesis), which describes the current state of the `SmaKi`, firstly the activated sensors (in this case some floor sensors and one motion detector) which are also easily identified in the previous Figure 20. In the “Association layer”, the one in the middle, are depicted the own detected items from the `SmaKi` (like the coffee machine, the table, the fridge or the bookshelf). The last part composing this graph is the “Representation layer”, in which, in this case, the “objects” represented in the first part are finally recognized as persons. Furthermore the recognized scenarios occurring at that are depicted to; in the current picture, the scenario “Meeting” and the scenario “Person takes Magazine”.

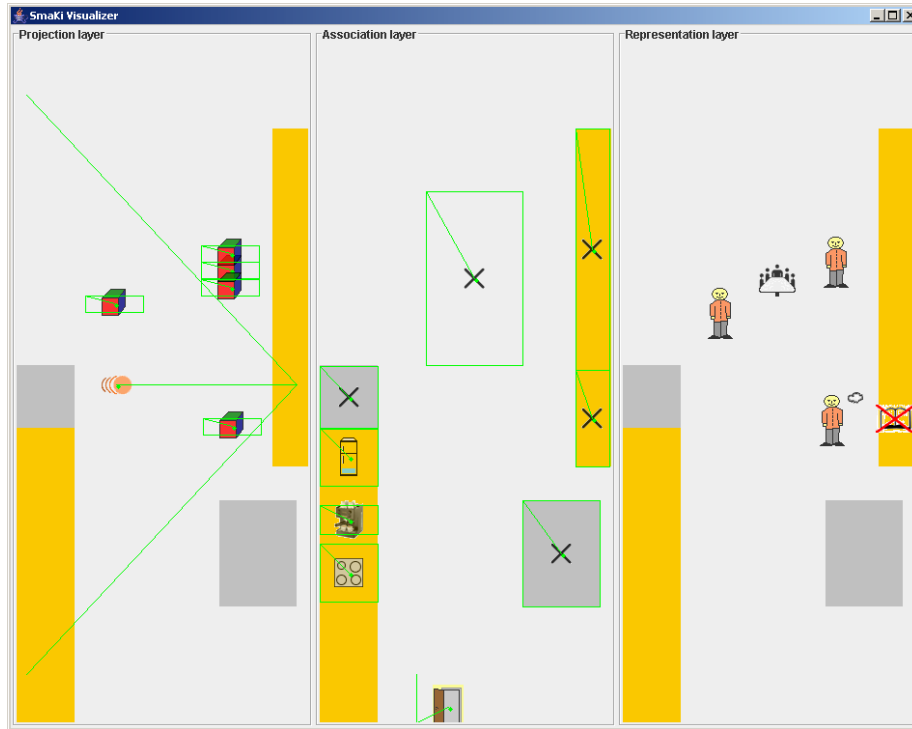


Figure 67: *SmaKi Visualizer*. Scenarios Meeting and “Person takes magazine” detected.

4.4.4 Scenario: Child near hot stove

The recognition of the scenario “Child near hot stove” was already considered in the ARS-PC, as well as the previous explained scenario “Child makes coffee”, this scenario is considered in the endeavour to avoid dangerous situations in which children are involved. There were mentioned in section 2.1.4 the requirements for the generation of this scenario: With the addition of the ARS-PC Vision to the whole system, the ARS-PC gets greater support in the detection of this scenario. The way ARS-PC Vision gives this support is by means of the adult-child distinguish application explained in section 4.4.1, which combined with the generation of the symbol “Hot stove” gives the basis for the scenario in matter. If a person in the SmaKi recognized as child is near the stove detected as hot, the scenario “Child near hot stove” is consequently generated.

4.5 Integration in ARS-PC system

Until now it has been always talked about the ARS-PC Vision as a system which, by means of the information obtained by the ARS-PC and the complementary information provided by the computer-vision layer, is able to recognize different situations and scenarios, generate the

corresponding symbols and send them back to the ARS-PC system; all this without considering the different mechanisms that involve the ARS-PC system to achieve the end applications. Actually the intention of this thesis is to build an entire unique system, nevertheless with the whole system explained throughout this work the remaining part concerning to the ARS-PC system procedure needs some upgrades for the complete integration of both systems.

It was already discussed in section 3.4 as part of the system design, the necessity of a whole set of new functions to communicate the system developed in this thesis with the previous one installed at the SmaKi. Furthermore it was also necessary additional modules to treat and manage the symbolic information. The reason of this fact derived of the disparity of the code languages between the previously implemented ARS-PC and the software-package chosen for the development of the computer-vision layer composing the ARS-PC Vision, the OpenCV. Hence, the ARS-PC Vision is in essence an external module to the ARS-PC, interconnected with a net-interface, which procedure reaches its limit when the symbol message is sent in the ARS-PC direction. After that, for the end-integration of both systems, it is necessary the development of some new methods and functions to face the generation of the different symbols within the different levels, until completing the generation of the supposed recognized scenario.

To carry out this task is briefly described in section 4.5.1 the way the ARS-PC is structured, a break down in which the different modules that compose this system are enumerated. Finally to end the entire implementation of the system and achieve the end-integration of both systems, the procedure to follow is explained in section 4.5.2, discussing the different modules in play.

4.5.1 ARS-PC structure

It was already described in section 2.2.3 the way the ARS-PC system is connected with the other components composing the whole system, the Octobus, the Sensor Database and the Symbol Database, as well as the connection related with the ARS-PC Vision. The applications of that components were discussed, furthermore the possibilities they provide for the development of the different running modes were explained in section 3.2; the ARS_Live and the ARS_Database. The first procedure works in real-time, recognizing the situations occurring constantly in the room; the second one is developed for tests purposes, no presence in the SmaKi is necessary, the information of the scenarios – in essence the sensor data – are taken from the Sensor Database. In Figure 68 are represented once again the mentioned connections, concretely for the ARS_Live configuration (observe that the ARS-PC considers in this case the use of the Octobus and the Sensor Database at the same time), this time with all the packages composing the ARS-PC system. Following the specifications given in [Ric07] are described next the implemented software components that compose the ARS-PC system:

msf_live – This package generates microsymbols from the incoming messages, received from the Octobus and also from ARS-PC Vision. The prefix “msf” derives from “Microsymbol factory” and is used in the real-time configuration, to convert sensor data y microsymbols.

symbols – The software-package `symbols` is entrusted to manage the different microsymbols in order to create new snapshot-symbols, representation-symbols and scenario-symbols.

fis – This package contains all functions that deal with the decision making methods. This package is called “fis” as reference to the Fuzzy-Interference-System.

ars_graph – The generation of the connections among the different symbol levels in the graphical representation of the ARS-PC is carried out by a set of functions that compose this module (see Figure 20).

smaki_vision – This package shows the recognized symbols in a simple graphic layer as representation of the SmaKi.

applications – This package contains applications which were defined for the ARS-PC system and provides a graphic representation for all them.

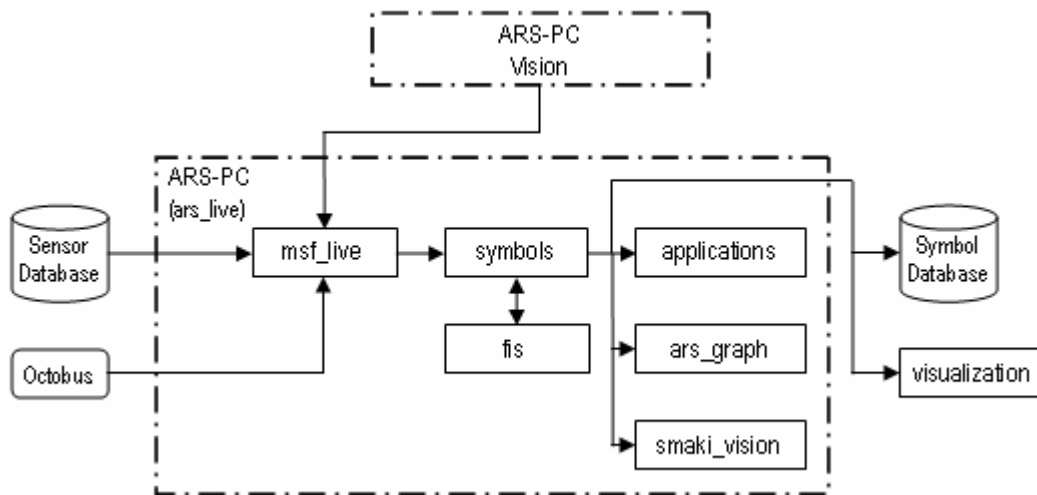


Figure 68: ARS-PC system break down with interconnections in ARS_Live configuration [Ric07]

The ARS_Database configuration uses all the packages mentioned above as exception for the `msf_live`, but instead, there is another module called `msf_database`, which extracts the sensor data from the Sensor Database and generates the corresponding microsymbols; of course the Octobus does not take part in this configuration.

4.5.2 Integration procedure

Section 4.5 was introduced referring to the limit where the ARS-PC Vision end its procedure in the whole process of the scenario recognition, as it makes the corresponding detections and sends the NSM to the ARS-PC (similar processes were explained in sections 4.4.1, 4.4.2, 4.4.3 and 4.4.4). Taking the previous section into account, is deduced that the module that will treat the messages sent by the ARS-PC Vision is the `msf_live` – in the case of the ARS_Live configuration. That means that, for the end integration of both systems, the first module that must be upgraded in the ARS-PC is the `msf_live` package, in which new functions to process the incoming NSM from the ARS-PC Vision must be considered.

After that, once the new functions are integrated in the `msf_live` package, it is necessary to reconsider the next step following the ARS-PC structure, the conversion of one or more microsymbols in snapshot-symbols; these symbols in turn must establish new connections to the representation layer and finally these symbols have to make the corresponding contribution to the generation of the scenario-symbols in the upper level. As part of this procedure, are two facts which are important to remark. On the one hand the integration of the ARS-PC Vision – as well as any another new source of information – to the ARS-PC, may involve the recognition of new features of the environment, what implicates the consideration of new symbols in the ARS symbol alphabet (see section 2.1.3) or even the adding of new properties to the already considered symbols. On the other hand, with the possible new redundant information, would be suitable to reconsider the implemented decision making methods contained in the `fis` package; this would achieve the optimal generation of the symbols among the different levels. Finally, the graphical layer, composed by the packages `applications`, `ars_graph` and `smaki_vision`, must be complemented with the new functions and the new added symbols (as it is depicted in Figure 67 where the generation of the scenario “Person takes magazine” is represented with a “crossed magazine” in the Representation layer).

Chapter 5 Results and Further Work

The aim of this work was the design and implementation of a computer vision system, based on the ARS model, as well as the proper integration of this system in the previous existing ARS-PC system. The designed system enjoys a full two-way communication with the ARS-PC, as if they were the same module and by means of different computer-vision applications, properly integrated with the rest of the sensors, it is able to provide support and enhancement in the task of scenario recognition.

The system was successfully implemented with two different operation modes. One of them works in a parallel way to the *ARS_Database*, which recognizes situations and scenarios that are the result of the reproduction of previously stored sensor-data. The other application receives information in real-time from all the sensors, even so from the cameras and thus the system at hand carries out the situation and scenario in real-time recognition, providing a constant control of the situation happening in the SmaKi.

In section 5.1 are evaluated the results obtained of the end release of the system, by means of the analyze of the weak points and its strengths for each of the application developed in this work. The following section, the one which encloses this thesis, composes a description about the guidelines suggested to follow in further works, lacks as well as upgrades that should be considered in order to achieve consistency in the following projects.

5.1 Results

There are two important facts to consider in order to evaluate objectively the obtained results of the system implementation. The first question is actually consequence of the main target for which this system was developed, the integration of the system with the ARS-PC. The system integration was brought up to demonstrate the profits of developing applications using all the available sensors, floor sensors, motion detectors, cameras, etc., thus some of the applications developed in the ARS-PC Vision start with results obtained by the ARS-PC. Therefore the different miss-detections produced in this ARS-PC system are dragged into the system developed in this work and of course returned back to the first system, which is the one entrusted to carry out the generation of the recognized scenarios. The second question to

take into account is that the knowledge of computer vision applied in this work was enough to prove the advantages of providing the whole system with a computer-vision system, but on the other hand they resulted sometimes insufficient as the results given by the ARS-PC Vision are not always optimal. Hence, taking these two considerations into account, the main functions and applications are valued next.

Function: Camera calibration – The camera calibration explained in the context of this work, is understood as a set of cameras configurations, different depending on the application in matter, which facilitate the proper operation of the applications, in sense of providing some coordinates or references to fulfil the requirements of the corresponding functions. There are three different kinds of camera calibrations developed in this work, a dynamic and a manual calibration specific for CAM1, and another one for CAM2.

The dynamic calibration represents a suitable tool for the camera configuration since the system calculates by itself the actual position of the device. However in the moment that the reference used for the self calibration – which in this case is the door of the SmaKi – is not well recognized, the calibration is not successful anymore. Because of the great dependency of other functions on the proper calibration of the camera, the use of the dynamic calibration in the system was discarded. Therefore the system was provided with the manual camera calibration function, which only depends on how accurately the measures are taken. This way once the camera is well calibrated, there is no reason to find an error produced by this function. The calibration implemented for CAM2, as well as the manual calibration for CAM1, works without problems, it only needs the proper focusing of the camera the first time it is installed and thus the control of the bookshelf is ensured to be successful.

Application: Adult/Child distinction – The aim of this application is to distinguish adults from children in order to avoid potential situations of danger where children are involved. The way this application is carried out is by means of the estimation of the corresponding height of the detected persons in the SmaKi. This application, the way it was implemented, does not only involve the right operation of several functions within the ARS-PC Vision, but also the proper working of the ARS-PC system and an accurate in-time synchronization between both systems. To analyze the weak points of this application it is important to break it down into the different tasks it is involved in. The first fact to consider is the proper calibration of the camera – the type of calibration selected for this application in order to drag the fewer possible errors was the manual calibration. Once the system is running the trigger of this application is the activation of the floor sensors. The ARS-PC must process the information and send the results to the ARS-PC Vision – taking into account that the Octobus facilitates the sensor values each 70ms. For the right calculation of the height of the detected person, the ARS-PC Vision needs to carry out the rest of the operations right in the same moment and place where the floor sensors were activated, otherwise the rest of the operations will be made of incoherent information. After the sensors are activated, the face detection is executed what normally takes around 50ms until the system detects a face, of course when the person is facing the camera and the conditions of illumination are favourable, otherwise

no faces will be detected and the whole process would stop at this point without giving any result. Furthermore, the entire floor in the SmaKi is not covered with floor sensors, and neither does the camera field cover the whole room. In regions where one of these two conditions is fulfilled the height estimation will not be satisfactory. Considering that all the steps previously mentioned are accomplished successfully the system will carry out the height calculation by means of some geometric estimations.

The different tests carried out for this application were manifold, however unfortunately no test was made with a child – what in height matter could be a person under 1,50 metres – used as target for the *height estimation*. Figure 69 is the result of a 114 samples test, made for a 171cm person moving all around the room – values are only obtained there where floor sensors are available, they are also limited to the region which fits into the camera field. A meaningful value that is deduced of this test is the resulting mean, which differs less than 1cm of the real size of the target. Taking this fact into account are suggested some possible developments or upgrades for this application as part of the further work related with computer vision in section 5.2.

Height Estimation test for a 171 cm person

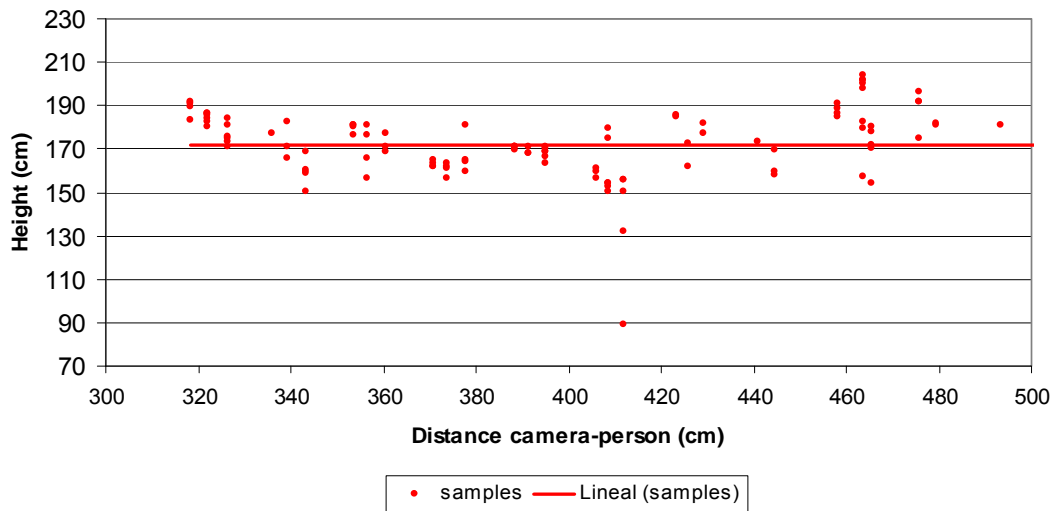


Figure 69: Dispersion graphic resulting from the *height estimation* test.

In Table 1 are resumed the different error percentages obtained as result of the mentioned test. In this table is shown the error magnitude as a difference to the real height of the target. It is divided in different intervals and the corresponding frequency is calculated in relation to the 114 samples generated. The table shows that the ~52% of the samples taken involve an error less than 10 cm and the ~88% an error less than 20cm.

Table 1: Error percentages relation obtained from the *height estimation* test.

ERROR MAGNITUDE; $ e $	SAMPLE PERCENTAGE
$ e < 5$ cm	28,08%
$5 \text{ cm} \leq e < 10$ cm	23,68%
$10 \text{ cm} \leq e < 15$ cm	23,68%
$15 \text{ cm} \leq e < 20$ cm	12,28%
$20 \text{ cm} \leq e $	12,28%

Scenario recognition: Person takes magazine – The scenario “Person takes magazine” represents the situation in which a detected person takes a magazine from the bookshelf. The scenario in matter represents a new situation for the ARS model, since it was not able to be recognized by the ARS-PC only with the previous sensors available. Therefore this scenario involves the consideration and consequent definition of new symbols in the ARS model.

This task, the way it is implemented, should be successfully achieved by the system without any problem in the most of the cases. Only two things were concluded to be the reason for miss-detection. The trigger in the generation of this scenario is the right detection of the person and its relative position to the bookshelf in which the camera is focused. If the ARS-PC system does not provide the proper detection of these two facts, it could result in a wrong generation of this scenario, or even result in a situation completely ignored by the ARS-PC Vision. On the other hand some difficulties related to the image processing can appear. If any of the magazines in the bookshelf has a similar colour as the bookshelf, it is very probable that the system would not detect the change when they were taken from its original place, since the difference between the two situations – the before and the after taking the magazine – could result negligible for the system. This situation is easily susceptible to be aggravated in situations with lack of light or strong variations of illumination. As exception of these possible adversities, this application is achieved with a great grade of success, tested in several scenarios combining all the variables in play.

Application: Child safety – For this application there were two scenarios taken into account, “Child makes coffee” and “Child near hot stove”. For the generation of any scenario were a child is involved the application “Adult/Child distinction” results indispensable. Hence, due to the way the system is implemented, if there are errors in the height detection, it is possible to detect a child instead of an adult or vice versa, what would result in a wrong generation the possible scenarios related. Nevertheless, if the person is a child and it is well recognized, the system should generate both scenarios without any problem. Moreover, although the SmaKi already equips a vibration at the coffee machine, the ARS-PC Vision provides support to the recognition of the “Child makes coffee” scenario, since it recognizes – when the illumination conditions are favourable – if a cup is set in the coffee machine.

In order to prove the scope of this application different test were made. Six different cups, with different colour and brilliance properties were used to prove the reliability of the system. Depicted in Figure 70 in clockwise, white (matt), white (brilliant), brown, grey, red and blue.

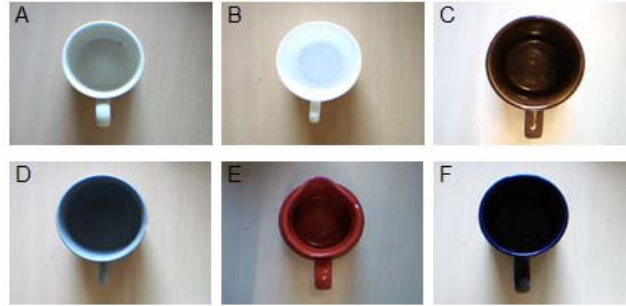


Figure 70: Cups with different colour properties used in the *cup detection* test.

The system was applied for each cup in different conditions of light, taking into account that the surface belonging to the coffee maker, where the cups are placed to prepare the coffee is a metal surface – and not the one of Figure 70 – which makes dizzy reflections in the image. The results obtained according to the different methods implemented are shown in Table 2.

Table 2: Relation of detectable cups depending on the light conditions

CUP	COLOUR	LIGHT CONDITIONS	
		ARTIFICIAL LIGHT	NATURAL LIGHT
A	white (matt)	NOT detectable	detectable
B	white (brilliant)	detectable	NOT detectable
C	brown	NOT detectable	detectable
D	grey	detectable	detectable
E	red	detectable	detectable
F	blue	NOT detectable	NOT detectable

Several tests were carried out, with different cups – different colour properties – and different illumination conditions; these are resumed in the twelve combinations shown. Table 2 shows the complexity this application involves; some cups – depending on its colour – are detected only with natural light and others with artificial light. There are also colours, like the blue one tested, that are not detectable under any circumstances. The results obtained by the implemented system, were not optimal for all the cases, nevertheless this system represents support to the vibration sensor at the coffee machine in case of failure.

Furthermore, apart from the two main facts mentioned to consider at the beginning of the section, another barrier was found through the development of this work. The libraries used for the development of the computer vision applications were the OpenCV; they were chosen since they are open source code and moreover they are probably the current computer-vision libraries with the wider range of functions. They are even a suitable tool for the first steps in

computer vision due to the great variety of samples and the clearly definition of the functions composing these libraries. Nevertheless, it was mentioned when these libraries were introduced (see section 2.3.2), that the OpenCV is composed by a specific graphical interface called HighGUI, which is just an addendum for quick software prototypes and experimentation setups. This last consideration involved the investment of many hours in trying to solve some incongruities derived from this graphical interface. There were two problems concerned to the HighGUI which affected most to the development of the system, the incapacity of this interface to configure the camera resolution and the deficit to interpret the (FourCC) codecs available in order to achieve an optimal video recording.

The problem concerning to the video resolution results ill-fated, since the image resolution is a crucial fact when the application is matter requires of image processing. The HighGUI not only does not compose any function to change to video resolution, but also sets the video resolution in an arbitrary way. These conclusions are the result of having tested these libraries with several cameras from different brands and different image resolutions. That means that the video resolution achievable with the cameras used in the implementation of this work, was never the highest provided by the different cameras. The other problem in matter results from the incapacity of the HighGUI to load all the video codecs available in the system. This question derives in the necessity to stick to the codecs detected by the OpenCV; the chosen codecs through the implementation were not the optimal – in sense of providing a great quality-compression relation – available in the system.

5.2 Further Work

Since the application field of this work, the computer vision and ARS in general, is very extensive, there are various aspects which are worth further considerations. The different aspects described next are deductions arisen of the implementation of the system, possible developments or application enhancement.

Computer vision

Computer Vision is a discipline which is in the first phases of its development. Nevertheless the possibilities this science can provide are already countless. This work has implemented only some of the basic tools of computer vision, and even so demonstrates the support it represents to the scenario recognition. Through the design of the system developed in this work, different tools were mentioned, which could be useful for the applications this thesis is intended for. Next are described the possible improvement of the implemented functions and the direct application of the tools previously mentioned.

One of the applications for which this work was intended is the *person tracking*. The mechanism used to detect persons was the *face detection* due to requirements deduced of the design of the system. However, there are many other procedures to detect a person and

actually to improve the task of *person tracking*, like it could be the detection of the human body shape or the motion detection by means of background subtraction. The way the ARS-PC Vision knows the number of persons in the room is by means of the results provided by the ARS-PC system, and not due to the detected faces in the room. Another proposal in order to enhance this task would be the assembly of a camera at the ceiling to count the different “heads” in its view field. Combining shape distinction and background subtraction could be the solution. A part from that, it is important to remark, that throughout this work it was always referred to *face detection* and not to *face recognition*. The face recognition entails the face detection but it even involves the development of complex algorithms in order to recognize the detected face among supposed “stored faces” in a supposed “face-database”. The development of this tool within the ARS-PC could bring the system a step further; hence the system could have a control of the persons which frequent the SmaKi. Is in this matter where the system leads to the critical issues of privacy, therefore this application was not implemented.

One of the tasks that involved much endeavour through the development of this work, and nevertheless resulted not as accurately as expected was the *person height estimation*. This question would not implicate the dependency of so many variables – what actually represents its weak point – if the procedure used for this task involved the use of stereocameras, which provide the system in matter with a 3D vision and thus a perception of the metrics within the image [DZ00,JH00]. Another possible procedure to improve the results obtained in this application keeping the current configuration with only one camera is described next. It was mentioned in the previous section 5.1 the good approximation of the mean resulting in the – 114 sample – adult/child distinction test. Taking advantage of this fact arises the possibility to develop of a *dynamic* estimation, if the estimated height is the mean resulting of all the values obtained during the person tracking. The more values the system consider, the more accurate the estimation.

Due to the sometimes imprecise detections achieved by some of the applications implemented, arises the necessity of an improvement of the algorithms developed or even a greater study of the available possibilities in matter of computer vision, in order to provide the system with a robust vision system that worth further development in this direction.

Integration

The integration of the computer vision – as well as any new art source of information – in the ARS-PC system, can facilitate the recognition of new situations and scenarios that are not considerate in the ARS model, this derivates in the necessity of defining new symbols or even the adding of some properties to the existing symbols. Examples of this fact are the necessity of a new property *Height* to the symbol *Person* or the definition of the new scenario *Person takes magazine*. Considering the further implementation of new applications the ARS system should develop methods to automatically define and create symbols – real-world grounded and understandable for humans – by itself.

Furthermore with the adding of new data sources and the increase of redundant information, arises the necessity of an upgrade of the current decision making methods developed in [Ric07]. An example of this fact results of the “cup detection” application developed through this work, which provides support to the previously – at the coffee machine – installed vibration sensor. Both devices, vibration sensor and camera, provide the same information – i.e. coffee is being done – through different mechanisms. Thereby the concept of redundant data introduced by the ARS is achieved, nevertheless, since the information is more sophisticated, more sophisticated decision making procedures should be developed too. The reliability of all the information at hand must be evaluated, considering not only external – to the devices – conditions, but also the current intrinsic states of these devices. For example, the information provided by the cameras loose reliability with the lack of light, in this case the resulting data generated by the vibration sensor – in the case of the coffee machine – should gain priority. This consideration should be applied always if redundant information provided by different devices is available.

The current ARS-PC system is the result of a series of works which started with the proposal of [Pra06] and ended thus far with the implementation of the ARS-PC Vision. The thesis at hand is the continuation of the work carried out by [Ric07], who not only did he define most of the current existing symbols of the ARS model but also enhanced it in a great extent recognizing scenarios with the integration of a decision making system based in *Fuzzy Logic* [Ric07] in the recognition of situations. At the same time this work was the continuation of a previous thesis carried out in [Goe06] which was the precursor of the design and implementation of the ARS real prototype at the SmaKi. Goetzinger designed the layout of the SmaKi – sensor network distribution and sensor hardware interface – and based on the ARS model described [Pra06] developed the first software composing the ARS-PC system. The series of this works involve a positive balance, what is reflected in the clearly improvement of a system which achieves better and better the initial purpose what it was intended for. However it is necessary to remark that these three systems, those developed in [Goe06] and [Ric07] and the one developed through this thesis, were carried out in three different periods of time. This question derived in some incongruities between the two first systems, what involved sometimes difficulties in their comprehension, as well as a possible dragging of any of those incongruities into the system at hand. A proposal to upgrade the ARS-PC system is the fusion of these three systems as part of a new thesis that deletes the possible deficiencies derived from the “lack of synchronism” among these works and optimizes the system by taking advantage of the proper combination of them.

Glossary

ARS	<u>A</u> rtificial <u>R</u> ecognition <u>S</u> ystem
2D /3D	Two <u>D</u> imensional /Three <u>D</u> imensional
ARS-PA	<u>ARS</u> - <u>P</u> sychoa <u>n</u> alysis
ARS-PC	<u>ARS</u> - <u>P</u> er <u>c</u> eption
CCD	<u>C</u> harge <u>C</u> ouple <u>D</u> evice
CMOS	<u>C</u> omplementary <u>M</u> etal <u>O</u> xide <u>S</u> emiconductor
COI	<u>C</u> hannel <u>o</u> f <u>I</u> nterest
DER	<u>D</u> istinguished <u>E</u> ncoding <u>R</u> ules
DSP	<u>D</u> igital <u>S</u> ignal <u>P</u> rocessing
EM	<u>E</u> xpire <u>M</u> essage
FourCC	<u>F</u> our <u>C</u> haracters <u>C</u> ode
HBM	<u>H</u> eart <u>B</u> eat <u>M</u> essage
ICT	<u>I</u> nstitute of <u>C</u> omputer <u>T</u> echonolgy
IPL	<u>I</u> ntel <u>I</u> mage <u>P</u> rocessing <u>L</u> ibrary
NSM	<u>N</u> ew <u>S</u> ymbol <u>M</u> essage
PIR	<u>P</u> assive <u>I</u> fra <u>R</u> ed.
RGB	<u>R</u> ed <u>G</u> reen <u>B</u> lue
SmaKi	<u>S</u> mart <u>K</u> itchen
SPD	<u>S</u> ize <u>P</u> ixel <u>D</u> istance
TCP	<u>T</u> ransmission <u>C</u> ontrol Protocol
UPM	<u>U</u> pdate <u>P</u> roperty <u>M</u> essage
USB	<u>U</u> niversal <u>S</u> erial <u>B</u> us
VGA	<u>V</u> ideo <u>G</u> raphics <u>A</u> rray

References

- [Big06] Bigun J.: *Vision with Direction – A systematic Introduction to Image Processing and Computer Vision*. Halmstad, Sweden. Editorial Springer, 2006.
- [Bra04] Brainin, E.; Dietrich, D., Palensky, P., Rösener Ch.; – *Neuro-bionic Architecture of Automation Systems - Obstacles and Challenges*. In Proceedings of the 7th IEEE African Conference, Gaborone, Botswana, 2004.
- [Cas96] Castleman, K.R.: *Digital Image Processing*. Prentice-Hall, Englewood Cliffs, New Jersey, 1996.
- [Cro05] Crownley, J.L.: *Computer Vision for Interactive and Intelligent Environment*, 2005, pages 97-108.
- [CY05] Chen, D.; Yang J.: *Online Learning of Region Confidences for Object Tracking*. In proceedings of the 2nd Joint IEEE International Workshop on VS-PETS, Beijing, China, 2005.
- [Die04] Dietrich, D.; Kastner, W.; Maly, Th.; Rösener, Ch.; Russ, G., Schweinzer, H.: *Situation Modeling*. In Proceedings of the 5th IEEE International Workshop on Factory Communication Systems (WFCS 2004), Vienna, Austria, 2004.
- [Dub00] Dubuisson, O.: *ASN.I – Communication between heterogeneous systems*. Morgan Kaufmann Publishers, Sept 2000.
- [DZ00] Dijvak, M.; Zazula, D.: *Accuracy of 3D motion tracking using a stereocamera*. University of Maribor, Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia 2000.
- [Elm02] Elmenreich, Wilfried: *Sensor Fusion in Time-Triggered Systems*, Faculty of Electrical Engineering and Information Technology, Vienna University of Technology, PhD thesis, 2002

- [FLP01] Faugeras, O.; Luong, Q.-T.; Papadopoulos, T.: *The Geometry of Multiple Images: The Laws That Govern the Formation of Multiple Images of a Scene and Some of Their Applications*. The MIT Press, Cambridge MA, London, 2001.
- [FP02] Forsyth D.; Ponce J.: *Computer Vision – A modern approach*. Prentice Hall, 2002.
- [Goe06] Götzinger, Sigfried: *Scenario Recognition Based on a Bionic Model for Multi-Level Symbolization*. Faculty of Electrical Engineering and Information Technology, Vienna University of Technology, diploma thesis, 2006.
- [Hol06] Holleis, Edgar: *SymbolNet – ein Application Framework für symbolische Kommunikation*. Vienna, June 2006
- [HS92] Haralick, R.M.; Shapiro, L.G.: *Computer and robot vision*. Addison-Wesley Publishing Co., New York, 1992.
- [JH00] Jähne, B.; Haussbecker H.: *Computer Vision and Applications – A Guide for Students and Practitioners*. Editorial Academic Press, 2000.
- [Kru06] Kruegle H.: *CCTV Surveillance – Analog and Digital Video Practices and Technology*. Editorial Butterworth-Heinemann, 2nd edition, 2006.
- [LK81] Lucas, B.D.; Kanade, T.: *An Iterative Image Registration Technique with an Application to Stereo Vision*. From Proceedings of Imaging Understanding Workshop, pages 121-130. Pittsburgh, 1981.
- [MK04] Medioni, G.; Kang S.B. : *Emerging Topics in Computer Vision*. Editorial Prentice Hall, 2004.
- [ORM01] *OpenCv Reference Manual*, © Intel Corporation, Order no. 123456-001. United States of America. 1999-2001.
- [Pra06] Pratl, Gerhard: *Processing and Symbolization of Ambient Sensor Data*. Faculty of Electrical Engineering and Information Technology, Vienna University of Technology, PhD thesis, 2006.
- [Ric07] Richstfeld, Andreas: *Szenarienerkennung durch symbolische Danteverarbeitung mit Fuzzi-Logic*. Institute of Computer Technology, Vienna University of Technology, diploma thesis, 2007.
- [Rös06] Rösener Charlotte: *Adaptative Behaviour for Mobile Service Robots in Building Automation*. Faculty of Electrical Engineering and Information Technology, Vienna University of Technology, PhD thesis, 2006.
- [RW96] Ritter, G.; Wilson J.: *Handbook of Computer Vision Algorithms in Image Algebra*, CRC Press, 1996.

- [SCGH06] Sebe, N.; Cohen I.; Garg A.; Huang T.S.: *Machine Learning in Computer Vision*, University of Amsterdam, HP Research Labs, Google Inc. and University of Illinois. Editorial Springer, 2005.
- [Sch07] Scheiber, Stefan: *Visualisation for the Smart Kitchen*. Institute of Computer Technology, Vienna University of Technology, 2007
- [Sin02] Sing-Tze, B.: *Pattern Recognition and Image Preprocessing*, Northern Illinois University, De Kalb, Illinois: Editorial Board, 2nd edition, 2002, pages 298-396.
- [SRT01] Soucek, S.; Russ, G.; Tamarit, C: *The Smart Kitchen Project – An Application of Fieldbus Technology to Domotics*. Vienna University of Technology, Vienna, 2001.
- [SS02] Shapiro, L.; Stockman, G.: *Computer Vision*, The University of Washington, Seattle, Washington and The Department of Computer Science, Michigan State University, East Lansing, MI, 2002.
- [Zan98] Zhang, Z.: *A Flexible New Technique for Camera Calibration*, Technical Report MSR-TR-98-71, Microsoft Research, first published 1998, edition 2002.

Internet Links

- [ARS07] *Project ARS – Institut für Computertechnik*, Technische Universität Wien. <http://ars.ict.tuwien.ac.at>
- [AIP07] *The American Institute of Physics – Physics Publications and Resources*, <http://www.aip.org> © 2007 American Institute of Physics.
- [BOE07] *Britannica Online Encyclopaedia*, <http://www.britannica.com>, © 2007 Encyclopædia Britannica, Inc.
- [CWF07] *Creative WebCam Features*, <http://www.creative.com> © 2007 Creative Technology Ltd.

- [ICT07] *Institut für Computertechnik (ICT)*, Technische Universität Wien:
<http://www.ict.tuwien.ac.at>
- [LWS07] *Logitech Webcam Specifications*, <http://www.logitech.com> © 2007
Logitech
- [MDN07] *MSDN*, Microsoft Developer Network , <http://msdn2.microsoft.com>
- [OCL07] *OpenCV Library Wiki*, <http://opencvlibrary.sourceforge.net>, started on
14th December 2006, edition 14th September 2007.
- [OED01] *Online Etymology Dictionary*, <http://www.etymonline.com> © 2001
Douglas Harper.
- [VAS06] *The VASE Lab*, <http://vase.essex.ac.uk/>, 2006, accessed on October 25th
2007
- [LTI05] *LTI-lib*, <http://ltilib.sourceforge.net>, 2005, accessed on October 25th 2007
- [VIG06] *VIGRA*, <http://kogs-www.informatik.uni-hamburg.de/~koethe/vigra/> ©
Ullrich Köthe, 2006, accessed on October 25th 2007.
- [JAV07] *JAVAVIS*, <http://javavis.sourceforge.net>, Robot Vision Group, Department
of Computer Science and Artificial Intelligence, University of Alicante,
2007, accessed on October 25th 2007 .