Project no: 269317

nSHIELD

new embedded Systems arcHItecturE for multi-Layer Dependable solutions

Instrument type: Collaborative Project, JTI-CP-ARTEMIS

Priority name: Embedded Systems

D7.2: Voice/Facial Recognition demonstrator - integration and validation plan

Due date of deliverable: M22 –2013.06.30

Actual submission date: M22 –2013.06.30

Start date of project: 01/09/2011 Duration: 36 months

Organisation name of lead contractor for this deliverable:

IPS Sistemi Programmabili, ETH

Revision [Final]

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2012)		
Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	**X**

Document Authors and Approvals

Authors		Date	Signature
Name	**Company**		
Paolo Azzoni	ETH		
Stefano Gosetti	ETH		
Ugo Bertacchini	ETH		
Luca Geretti	UNIUD		
Dimitris Geneiatakis	TUC		
Reviewed by			
Name	**Company**		
Approved by			
Name	**Company**		

Applicable Documents		
ID	**Document**	**Description**
[01]	TA	nSHIELD Technical Annex

Modification History		
Issue	**Date**	**Description**
V0.1	01/05/2013	First draft
V0.2	29/05/2013	ETH Update
V0.3	30/05/2013	TUC Update
V0.4	30/05/2013	UNIUD Update
V0.5	31/05/2013	ETH Update
V0.6	04/05/2013	TUC, UNIUD, ETH Update
V0.9	06/06/2013	Whole document revision
V1.0	26/06/2013	Final review

Executive Summary

This deliverable illustrates the second application use case planned in the context of the nSHIELD project. The use case is called "People Identification at the Stadium" and represents an evolution of the original use case "Face and voice recognition" and it has been introduced to better illustrate the capabilities and potentialities of the adopted technologies and for its relevance in terms of market interest.

In the context of this use case several technology prototypes have been and will be developed. These prototypes will contribute step by step to the implementation of a conjunct final demonstrator.

The following partners will contribute and cooperate to the development of the final demonstrator: ETH, TUC and UNIUD.

The document is structured as follows:

- *Chapter 1 and 2 provide a brief introduction on the "People Identification at the Stadium" use case.*
- *Chapter 3 illustrates the hardware and software architecture of the demonstrator.*
- *Chapter 4 provides an overview of the technologies adopted in the demonstrator.*
- *Chapter 5 presents the scenarios of the demonstrator.*
- *Chapter 6 presents the integration plan.*
- *Chapter 7 presents the validation and verification plan.*
- *Chapter 8 presents the conclusions.*

Contents

Figures

Tables

Glossary

Please refer to the Glossary document, which is common for all the deliverables in nSHIELD.

1 Introduction

This deliverable illustrates the second application use case planned in the context of the nSHIELD project. The use case is called “People Identification at the Stadium” and represents an evolution of the original use case “Face and voice recognition” and it has been introduced to better illustrate the capabilities and potentialities of the adopted technologies and for its relevance in terms of market interest.

In the context of this use case several technology prototypes have been and will be developed. These prototypes will contribute step by step to the implementation of a conjunct final demonstrator.

The following partners will contribute and cooperate to the development of the final demonstrator: ETH, TUC and UNIUD.

Considering the market requirements, the research and development activities during the first two years of the project have been mainly focused on the face recognition technologies. Face recognition can provide in a short time more exploitable results than voice recognition/identification technologies, which is currently still in a theoretical phase of research. For this reason, the integration and validation plan if focused on face recognition technologies. Also, the evolution of the use case follows this decision. Voice recognition/identification technologies will be addressed in the last year of the project and the integration/validation plan will be updated accordingly.

2 nSHIELD People Identification use case

The use case “People Identification at the Stadium” has been conceived to demonstrate and exploit the SPD functionalities and capabilities of the technologies developed in nSHIELD when applied to the real application context of stadium security.

The stadium security is currently managed, in the large part of this specific market, by traditional systems based on security cameras connected to a security control centre. The cameras are remotely controlled by the security personnel and the video streams acquired are saved on a data server, without any processing.

The most important limitation of this approach is the complete absence of real time automatic elaboration of the video stream, which could provide important information directly when the event monitored are happening: the current solutions allow only to analyse the video stream after the events occurred, only during investigations, and rely entirely on the security personnel expertise.

Furthermore, at the entrance of the stadium, at the turnstile, the people identification is still based on a ticket and it is very difficult to find more advanced solution for the identification.

The solution proposed in this use case introduces a new paradigm in stadium security management based on the concept of prevention, which is implemented using a proactive video stream analysis that allows near real-time people identification.

From a technology point of view, this approach is based on face recognition algorithms that are executed directly on the security camera that, in this way, is capable to identify people autonomously. This solution provides information when the monitored event is happening and doesn’t require the presence of the security personnel.

The solution is enforced by the use of smartcard based access delegation services and by a dependable framework for distributed computing. The use of smartcard provides an important alternative source of information related to the identity of people. The dependable framework for distributed computing introduces several advantages on the technical side, allowing the implementation of a distributed system of smart cameras that are optimized in term of efficiency, security, dependability and costs.

Following the paradigm based on prevention, the implementation of the proposed situation covers the security aspects related to identification both at the entrance of the stadium and in the stadium during a match. At the entrance, at the turnstile, the identity of people is checked in order to deny the access to criminals, unwanted people and untrusted people. In the stadium, on the tribunes, the system dynamically identify people that are responsible for behaviours not allowed on the tribune, clashes between supporters, throwing of objects, damages to the stadium, etc..

Figure 2-1 illustrates the scenario of people identification at the entrance on the stadium.

When the person is in front of the turnstile, the camera acquires the images of his/her face and extract the corresponding biometric profile. The system asks to insert the smart card in the specific reader and read the biometric profile from the smartcard in a secure way. The two biometric profiles are compared and the person is allowed to enter only if they match and if the person is not in the list of unwanted or untrusted people.

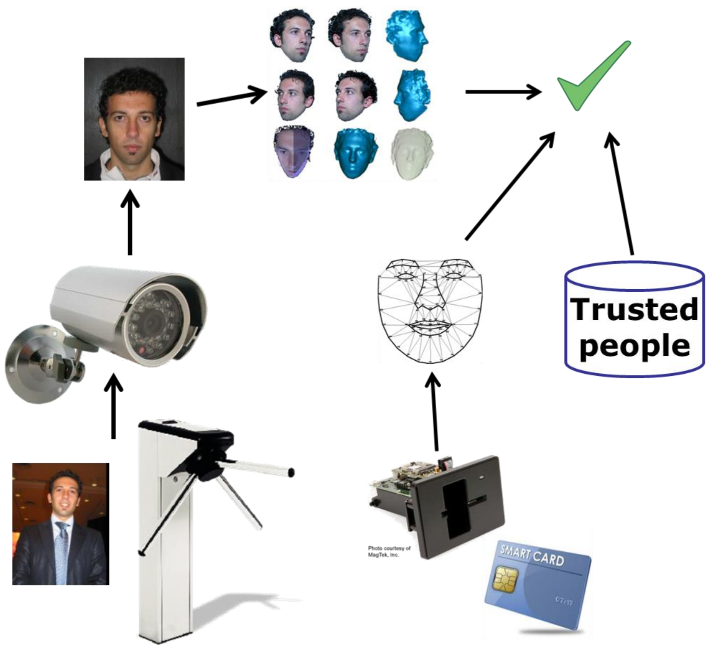


Figure 2-1: People identification at the stadium entrance

The security in the stadium during a match is monitored using the same technologies organized in a different way. A set of security cameras is placed around the playing area and monitors the tribunes. The position and number of cameras guarantee the coverage of all the tribunes. The security cameras are equipped with the face recognition software and are capable to identify people on the tribunes autonomously. The cameras can be hided with appropriate structures in order to be protected and remain almost invisible to the public on the tribunes. During a match, if there is a critical situation, the security personnel can guide one or more cameras, in order to point on the area of interest and obtain automatically the identity of the people present in that area.

The following figure illustrates the scenario of people identification in the stadium on the tribunes.

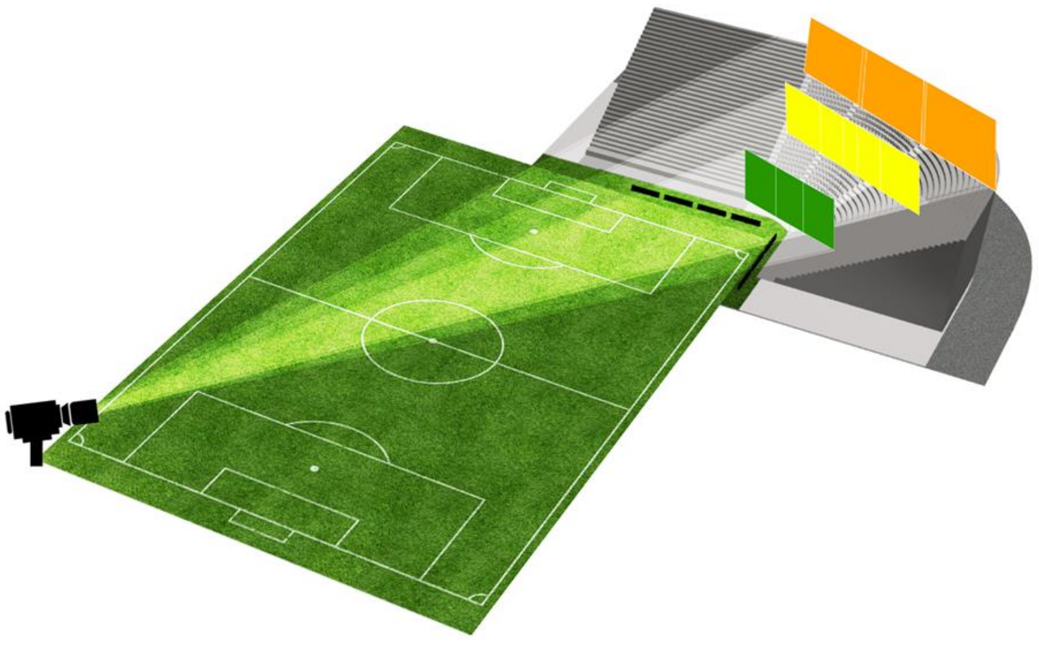


Figure 2-2: People identification in the stadium on the tribunes

3 People identification demonstrator reference architecture

This section contains a general description of the People Identification demonstrator and an outline of SW & HW architecture. Involved actors, components and interfaces among them are defined and will be described in the integration section.

3.1 Prototypes involved in the use case

The people identification use case is composed by a set of individual prototypes in the nSHIELD framework, which cooperate to create the corresponding demonstrator. They are enumerated in the following table with the associated Id code.

Table 3-1: Prototypes list

Id	Name	Author
06	Smart Card	TUC
07	Facial Recognition	ETH
14	Dependable Distributed Computation Framework	UNIUD
37	ETH SecuBoard	ETH

3.2 Hardware architecture

The people identification system is a distributed system, with a client server architecture, based on nSHIELD embedded nodes and a central server. The components of the system are connected through a local area network via cable or Wi-Fi. The following hardware components will be used to implement the demonstrator:

- ETH SecuBoard,
- IP cameras based on the ETH SecuBoard,
- smart cards,
- smart card reader,
- embedded computer,
- industrial PC,
- a personal computer.

The camera based on the ETH SecuBoard is responsible for the video stream acquisition, both in the scenario of the people identification at the turnstile and in the scenario of the people identification on the tribunes. The difference between the scenarios is only the optics mounted on the camera. The biometric profile of a person is extracted from the video stream acquired by these cameras.

The smart card is used as a storage medium for the biometric profile that is read from it using a smart card reader. The reader is connected via USB to the embedded board contained in the turnstile.

The industrial PC is “paired” with the embedded cameras and the turnstile and executes some modules of the face recognition application.

A personal computer is used for distributed face recognition or to simulate the central server.

The following figures illustrate the hardware architectures of the demonstrator in two different scenarios that will be illustrated in section 5.

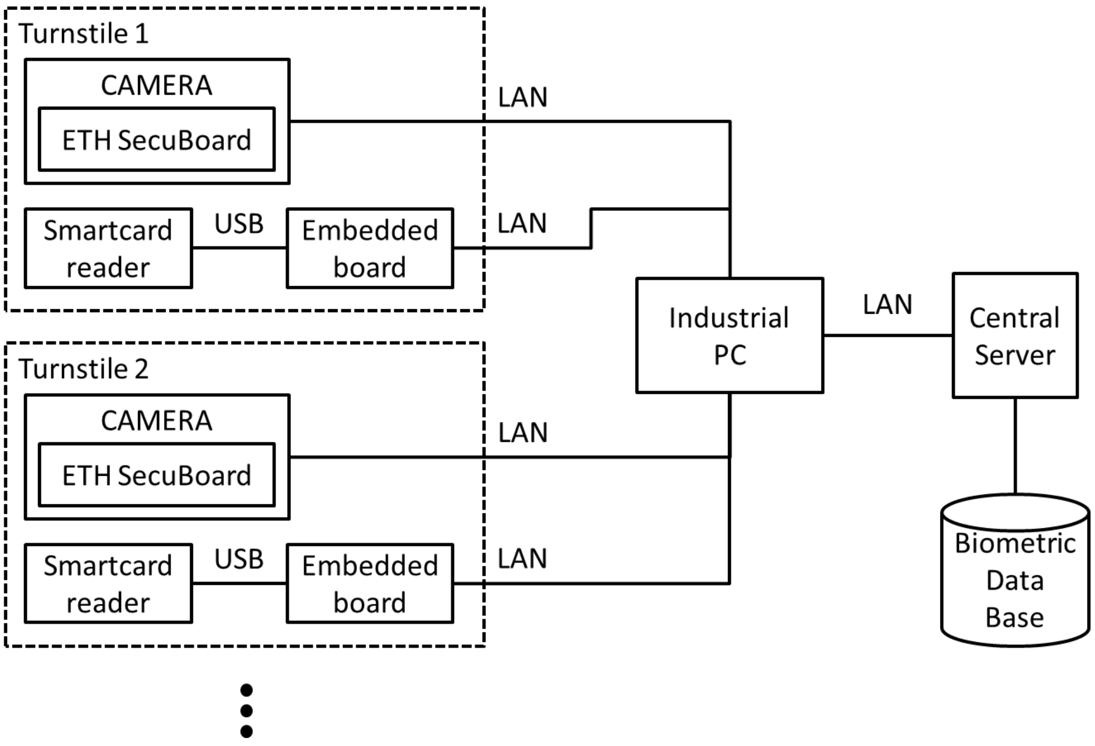


Figure 3-1: Hardware architecture of the people identification at the turnstile

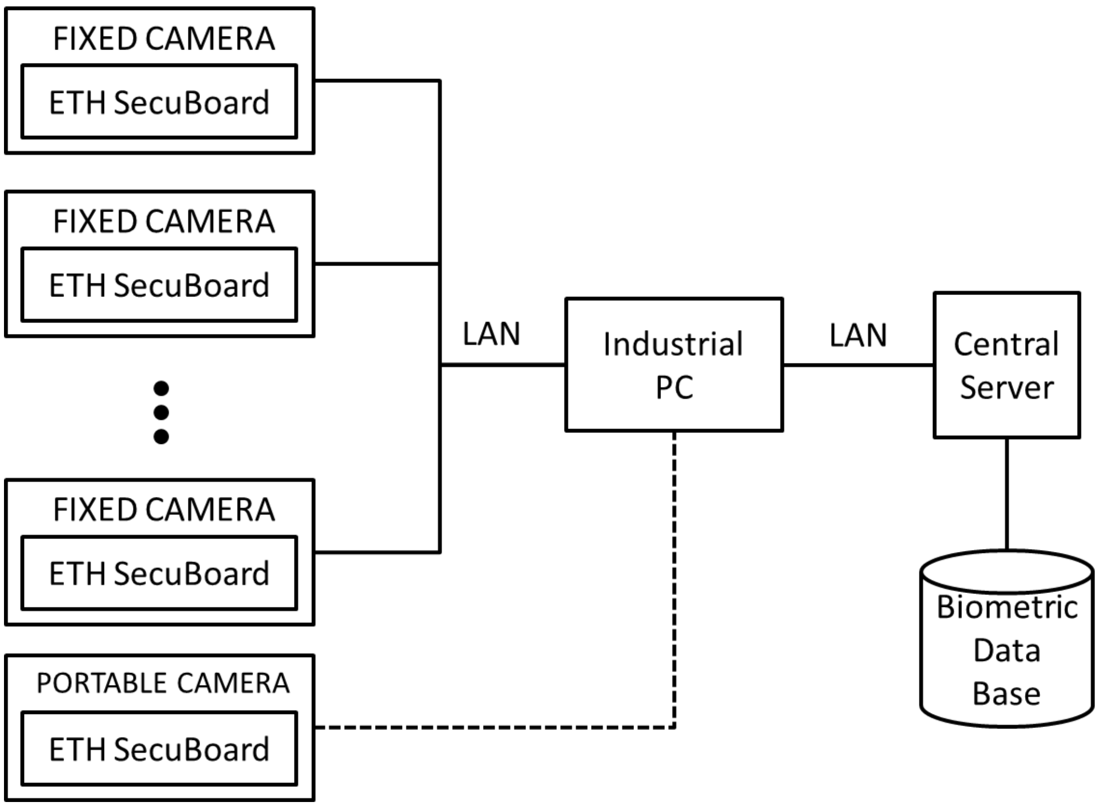


Figure 3-2: Hardware architecture of the people identification on the tribune

3.2.1 The ETH SecuBoard

The ETH SecuBoard is a custom board based on a system on chip (SoC) manufactured by Texas Instruments and based on ARM architecture: the DM816x DaVinci system on a chip. The board has been conceived specifically for video processing and provides natively hardware encoders and a digital signal processor (see the figure below).

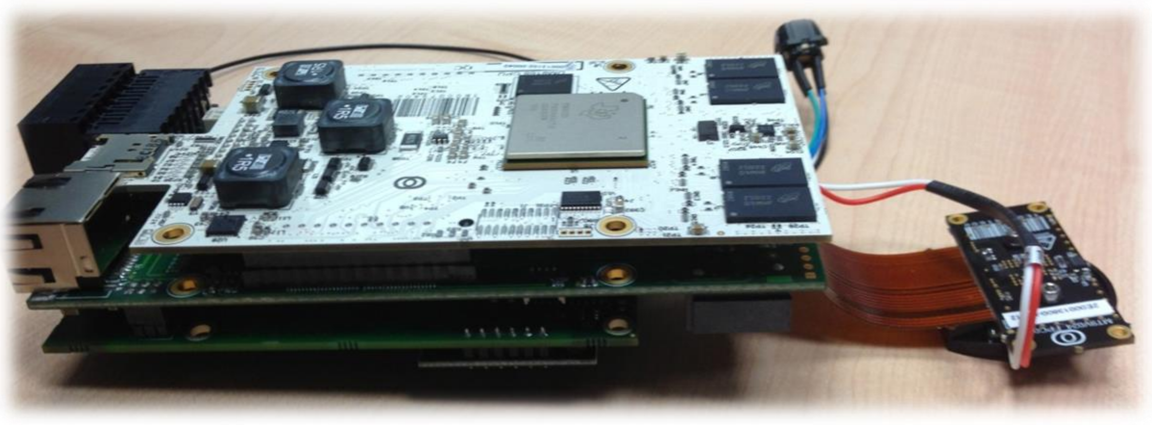
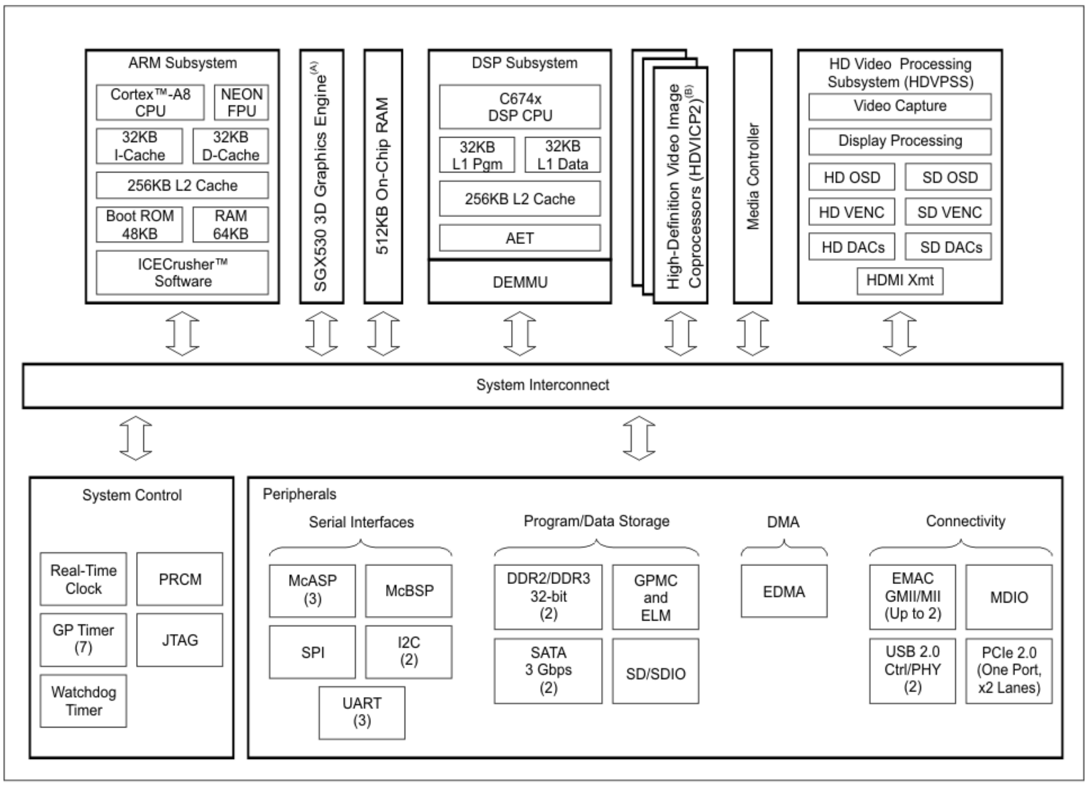


Figure 3-3: The custom embedded board based on TI SoC

The architecture of the board is described in Figure 3-4.



A. SGX530 is available only on the TMS320DM8168 and TMS320DM8166 devices.
B. Three HD Video Image Coprocessors (HDVICP2) are available on the TMS320DM8168 and TMS320DM8167 devices; two are available on the TMS320DM8166 and TMS320DM8165 devices.

Figure 3-4: The embedded board architecture

The block diagram of the components of the system on chip is described in the figure below. The figure illustrates also how the various units of the SoC cooperate together to provide high level imaging and video performance in a real embedded system board.

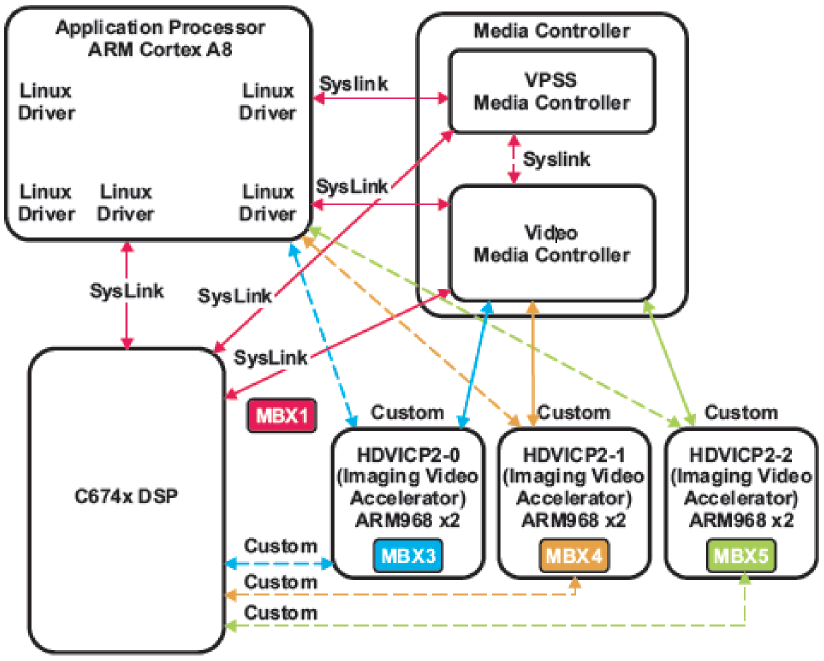


Figure 3-5: The SoC architecture

Further technical details can be found in deliverable D3.3, “Preliminary SPD node technologies prototype report”.

The camera based on the ETH SecuBoard will be developed specifically to contain this board and will provide all the rugged features of a standard surveillance camera for industrial and defence application.

3.2.2 The smart card and the smart card reader

The smart card reader that has been selected for the demonstrator is the SCR3310 V2 - USB Smart Card Reader. It supports every ISO 7816 Class A, B smart card. Any USB that can support this standard can be used instead of the selected one.

The manufacturer developed an SDK that provides demo, tools and PC/SC source code samples for development in VC++, Delphi, C#, VB.NET. The selected smart card reader is supported under Linux: drivers and development information can be found on the web site https://wiki.debian.org/Smartcards.

3.3 Software architecture

The software architecture of the “People Identification at the Stadium” use case is based on three main components:

- the face recognition software,
- the access right delegation system and
- the dependable distributed computation framework.

The following figure illustrates the software architecture of the demonstrator. The architecture refers to the scenario of the people identification at the turnstile, while for the scenario of the people identification on the tribunes the architecture is the same without the part related to the smart card security services (see section 5).

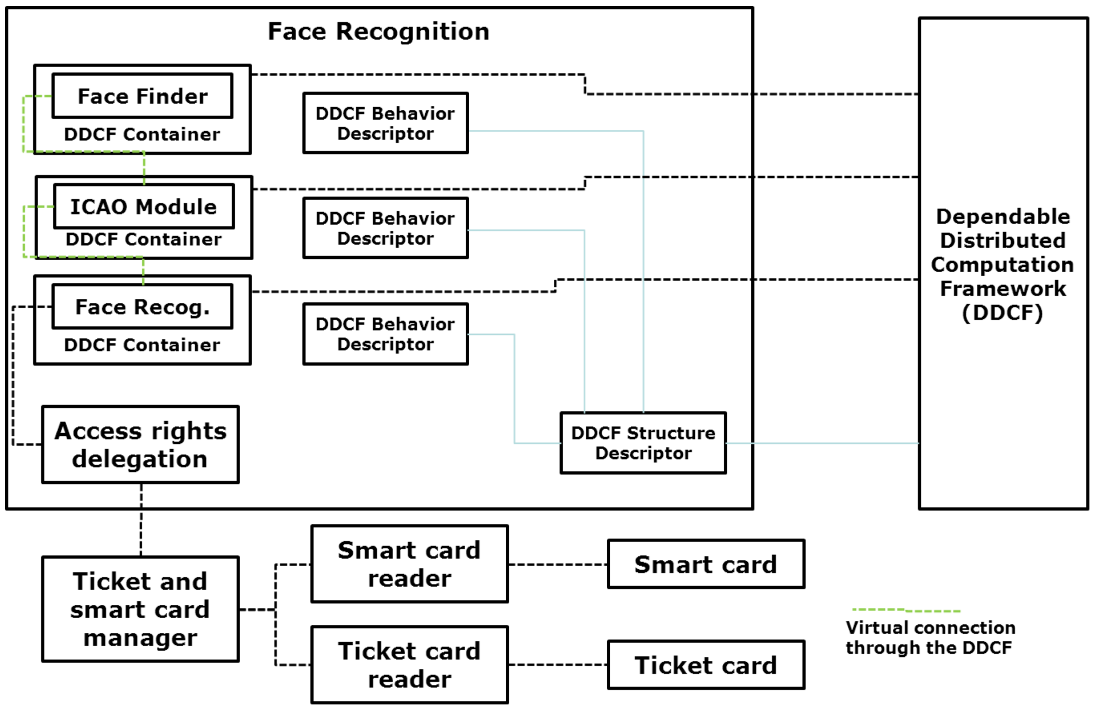


Figure 3-6: Use case software architecture

3.3.1 Face Recognition Software

The face recognition software is a modular application that implements an approach for face recognition based on the Eigenface method. This method is based on the idea of extracting the basic features of the face: the objective is to reduce the problem to a lower dimension maintaining, at the same time, the level of dependability required for such an application context. This approach has been theoretically studied during the nineties and has been recently reconsidered because it provides a good ratio between the required resources and the quality of the results and it is well dimensioned for embedded systems. Today, it is becoming the most credited method for face recognition. Further technical details can be found in deliverable D3.3, “Preliminary SPD node technologies prototype report”.

The Face Recognition Process

The core of this solution is the extraction of the principal components of the faces distribution, which is performed using the Principal Component Analysis (PCA) method. This method is also known in the pattern recognition context as Karhunen-Loève (KL) transform. The principal components of the faces are eigenvectors and can be computed from the covariance matrix of the face pictures set (faces to recognize). Every single eigenvector represents the feature set of the differences among the face picture set. The graphical representations of the eigenvectors are also similar to real faces (also called eigenfaces). The PCA method is autonomous and therefore it is particularly suggested for unsupervised and automatic face recognition systems.

The face recognition software has been developed using language C++, with C, C++, C#, Visual Basic API for Microsoft platform.

The recognition process that has been implemented is based on the following steps:

- face detection,
- face normalization,
- face extraction,
- template creation,

- template matching,
- identification.

The face recognition software is composed of a set of logical modules, which implement each of the logical steps of the face recognition process:

- Face Detection – This logical module finds the position and the size of each visible face present in the acquired image. This logical module corresponds to a real software module, the FF module.
- Face Normalization – This logical module identifies the features of morphological interest throughout the face area (eyes, mouth, and eyebrows), in order to properly rotate and scale the image. This logical module is a part of the software module called ICAO module.
- Feature Extraction module – This module selects the relevant features from the normalized image in order to maximize the robustness against environmental disturbance and noise, non-optimal pose, non-neutral facial expressions and variable illumination conditions. This logical module is a part of the software module called ICAO module.
- Biometric Template Creation module – The biometric template is implemented by a feature vector template that is created by this module. The extracted feature vector template is processed by a trained statistical engine and is reduced to a smaller one that optimally describes the user's identity. This logical module is a part of the software module called FR module.
- Biometric Template Matching module – The face recognition and the face verification processes are based on measurements of differences between biometric templates. This logical module is a part of the software module called FR module.

Finally, a classification logical module has been developed in order to provide the previous modules with a horizontal tool capable to offer the classification features required during the various steps of the recognition process. The classification module is based on methods for statistical classification of models and is capable to manage information partially compromised by noise or occlusions.

The logical modules are contained in software modules that are described in details in Section 6.1.

3.3.2 Smartcard and Access Rights Delegation

A smartcard is a tamperproof secure device resilient to physical attacks used to perform secure transactions. Smartcards are used in a plethora of applications that require security such as payment applications, healthcare, physical access control to mention a few. Smartcards can provide multiple security levels for sensitive data stored in them. For instance, a security key can be marked as read-only, while the read operation is accomplished only inside the smartcard. Even more the security key can be protected by a PIN to add one more security level. One of the main advantages of smart card solution is that all the sensitive operations are accomplished in the smart card rather than the terminal or application, which in many cases is not considered trustworthy. Smartcards among to others provide the following security services: message authentication code, encryption, identity validity, digital signatures, hash functions and secure key management.

The security services provided by smartcards can be exploited for delegation of access rights. In a network of offline trusted embedded systems, a node need to be able to authenticate another node requesting some privileges, but also to determine what – if any – privileges should be granted.

A real world example where delegation of access rights would be relevant is where the access rights issued by the central authority are valid only if the node seeking access has first interacted with another node. For example node: "Allow access to door B, but only if the visitor has first passed door A". In this example, door A is entrusted to verify that a correct passage has taken place (see Section 5.3).

The approach that has been selected adopts a construct called a path array. Nodes to which some authority has been delegated can read and write to the path array, thereby proving that a bona fide interaction has taken place. This information can then be used by the next node in the path array as part of the access control decision.

3.3.3 Dependable Distributed Computation Framework

The Dependable Distributed Computation Framework (officially named Atta, for short) is a middleware that allows applications to be run on multiple nodes in a distributed way. It essentially enhances dependability by managing nodes redundancy and also adds security both in the registration of the nodes and the transmission of data.

The role of Atta in the Face Recognition scenario is to add the cited features to the ETH SecuBoard prototype (prototype 37). In this context, the face recognition routines in prototype 37 are written as library objects that Atta is responsible to deploy, (possibly) compile and run on the nodes of the distributed platform.

Interfacing with Atta is mostly a design concern: an application must be (re)designed using a specific dataflow model. Each vertex of the application model contains code, while each edge represents a data transfer. The effort from the designer is therefore to think in terms of independent sections of code that interact with each other.

It must be noted that Atta does not force the designer to abandon his compilation and testing flows of choice: as soon as an application model has been designed, each code section can be built and tested in isolation using the preferred tools. With little adaptation, all sections can be linked together to test the whole application in a monolithic way. Consequently, while an “Atta-aware” design introduces additional concerns, it still accommodates as much as possible the existing design conventions.

4 People identification demonstrator technology overview

This section describes in details the various components and the relevant technologies involved in the People Identification demonstrator which have been identified in the previous section. A description of the following nSHIELD prototypes is provided:

- face recognition,
- smart card and access rights delegation,
- dependable distributed computation framework.

4.1 Face recognition prototypes

The Face Recognition prototype (prototype 7) and the ETH SecuBoard prototype (prototype 37) are the components of the use case “People Identification at the Stadium” that implement the face recognition technologies.

The face recognition prototype (prototype 7) is an all-in-one solution conceived to practically demonstrate the functionalities and potentialities of the face recognition system for people identification. It represents a proof of concept of the technologies adopted for the face recognition and it will be used to develop the final prototype, the ETH SecuBoard. This prototype belong to the former “Face and voice recognition” application scenario and has been developed during the first part of the nSHIELD project. During the second part of the project it will be finalized developing the embedded camera for face recognition that can be used in a real environment (prototype 37) and in the final demonstrator.

The two prototypes integrate with the smart card security services, introduced by TUC, and with the dependable distributed computation framework, developed by UNIUD.

4.1.1 Embedded System Based Face Recognition for People Identification

The automatic identification of people using a camera is an important topic in security and, recently, several new face recognition technologies have been proposed in the scientific community. These technologies have been evaluated and assessed in D3.1 “SPD node technologies assessment” and are based mainly on the following approaches:

- usage of three-dimensional (3D) scans,
- recognition from high resolution still images,
- recognition from multiple still images and
- multi-modal face recognition.

In addition, considering real applications, it is important to consider a set of multi-algorithms and pre-processing algorithms that can be applied to the previous approaches in order to correct the illumination and pose variations. The goal of these technologies is to significantly improve the performance of automatic face recognition, in particular when it is based on embedded system platforms that operate in real environment with uncontrolled conditions.

The assessment phase performed during the first part of the project allowed to:

- identify the most suitable recognition and identification algorithm,
- define an evaluation framework to ensure a high level of reliability during the recognition process.

4.1.2 The Recognition Technologies

The SPD approach identified in the assessment phase is the Eigenface method. This method is based on the idea of extracting the basic features of the face: the objective is to reduce the problem to a lower dimension maintaining, at the same time, the level of dependability required for this application context. This approach has been theoretically studied during the nineties and has been recently reconsidered because, considering the ratio between the required resources and the quality of the results, it is well dimensioned for embedded systems [1] and [2]. Today, it is becoming the most credited method for face recognition, [3] and [4]. The core of this solution is the extraction of the principal components of the faces distribution, which is performed using the Principal Component Analysis (PCA) method. This method is also known in the pattern recognition context as Karhunen-Loève (KL) transform. The principal components of the faces are eigenvectors and can be computed from the covariance matrix of the face pictures set (faces to recognize). Every single eigenvector represents the feature set of the differences among the face picture set. The graphical representations of the eigenvectors are also similar to real faces and, for this reason, they are called eigenfaces. The PCA method is autonomous and therefore is particularly suggested for unsupervised and automatic face recognition systems.

The recognition process is performed conceptually in three steps:

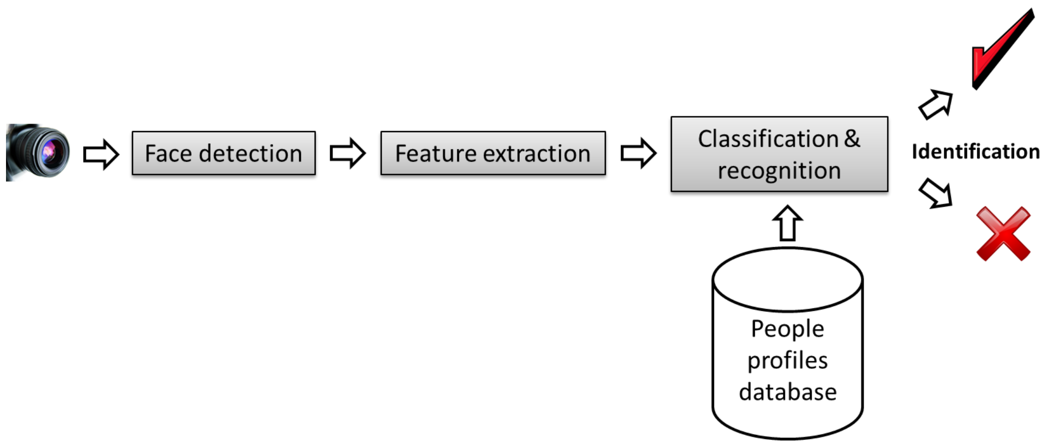


Figure 4-1: Recognition process main phases

The mathematical key concept on which the algorithm is based is that the eigenfaces are the principal components of the distribution of faces, or the eigenvectors of the covariance matrix of the set of face images. Starting from this idea the approach proposes the following steps:

1. The PCA method is used to find the eigen-vectors, called “eigenfaces”, of the covariance matrix corresponding to the generic training images.
2. The eigenvectors are ordered to represent different amounts of the variation, respectively, among the faces. Each face can be represented exactly by a linear combination of the eigenfaces. It can also be approximated using only the “best” eigenvectors with the largest eigenvalues.
3. Identification of the “face space”: the best M eigenfaces are used to construct an M dimensional space.
4. In the final step, unknown face pictures are projected on the face space to compute the distance from the reference faces. The weights describing each face are obtained by projecting the face image onto the eigenface.

4.2 Smart card and access right delegation

4.2.1 Overview

A smartcard is a tamperproof secure device resilient to physical attacks used to perform secure transactions. Smartcards are used in a plethora of applications require security such as payment applications, healthcare, physical access control to mention a few. Smartcards can provide multiple security levels for sensitive data stored in them. For instance, a security key can be marked as read-only, while the read operation is accomplished only inside the smartcard. Even more the security key can be protected by a PIN to add one more security level. One of the main advantages of smart card solution is that all the sensitive operations are accomplished in the smart card rather than the terminal or application, which in many cases is not considered trustworthy. Smartcards among to others provide the following security services:

1. Message Authentication code
2. Encryption
3. Identity validity
4. Digital signatures
5. Hash functions
6. Secure key management

4.2.2 Communication with Smartcards

Smartcards have the structure depicted in the figure below.

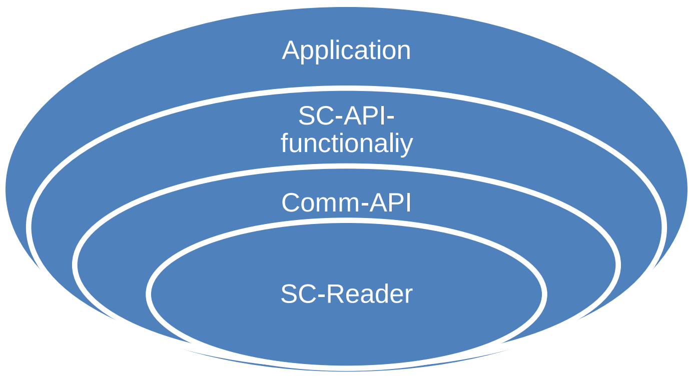


Figure 4-2: SmartCard communication structure

It should be noted that even in cases that smartcards do not provide a specific API for communication between the application and the smart card the communication with the can be accomplished by issuing direct command to the smartcard since the smartcards follows the ISO standards [10].

4.2.3 Secure services with smart cards

Depending on the type and the manufacturer the smartcards support a number of cryptographic features, including:

- On-card generation of symmetric keys and public key algorithms key pairs
- Digital signatures (based on public key algorithms)
- Symmetric encryption and decryption

- External authentication (host to card)
- Internal authentication (card to host)
- Message authentication code
- Hash functions

Further, smartcards enable protected mode for highly sensitive data, which requires commands to be authenticated and integrity protected either with symmetric or asymmetric keys.

4.2.4 Building Secure Communications

In order to build trust among different type of nodes on the nSHIELD architecture we can exploit the benefits of smart cards and the cryptographic schemes they implement. Considering, the nSHIELD architecture where decentralized components are interacting not only with each other but also with centralized ones, depending on the type of the device and the employed scenario; there is a need for integrating security and interoperability. In this context, we rely on [11] for building secure communication channels among different devices. We should mention that smart cards currently are used in various applications, where proof-tamper devices are need for the provision of security services.

4.2.5 Access Rights Delegation

Within the scope of D3.2 (WP3, Task 2) an approach to delegation of access rights has been investigated. In a network of offline trusted embedded systems, a node need to be able to authenticate another node requesting some privileges, but also to determine what – if any – privileges should be granted. A model for doing this has previously been developed by the project partner (Telcred), but this model assumes that all access rights are issued by a central trusted authority and does not support delegation.

A real world example where delegation of access rights would be relevant is where the access rights issued by the central authority are valid only if the node seeking access has first interacted with another node. For example node: “Allow access to door B, but only if the visitor has first passed door A”. In this example, door A is entrusted to verify that a correct passage has taken place.

The approach that was investigated uses a construct called a path array. Nodes to which some authority has been delegated can read and write to the path array, thereby proving that a bona fide interaction has taken place. This information can then be used by the next node in the path array as part of the access control decision.

The work was mainly carried out as a M.Sc. thesis at KTH, the Royal University of Technology in Stockholm.

4.2.5.1 Problem Statement

In an offline PACS (Physical Access Control System), there is no continuous exchange of information to verify and allow a user through a series of doors, whereas this is a common feature in an online PACS. Current offline systems are unable to force a user to follow a certain designated route, e.g. Room A should be accessed before entering room B. This project explores a model to enforce such a route, by using delegation of some authority from the main administrative system to the offline locks.

4.2.5.2 The Concept of “Path Array”

The developed artefact consists of a construct known as Path Array aka PA. Path Array is an array that can be one or multi-dimensional based upon the administrator requirements. PA consists of $\texttt{Lock}_{id}$ stored into each index of the array that needs to be accessed by the user in a sequence. Administrator is responsible to implement path array onto the user’s smart card before handing it over to the user.

`Ticket` that is stored in the flash memory of the smart card contains the PA. After the formation of mutual trust between the `Lock` and `Card`, `Lock` makes use of the remaining contents inside the `Ticket` for decision making.

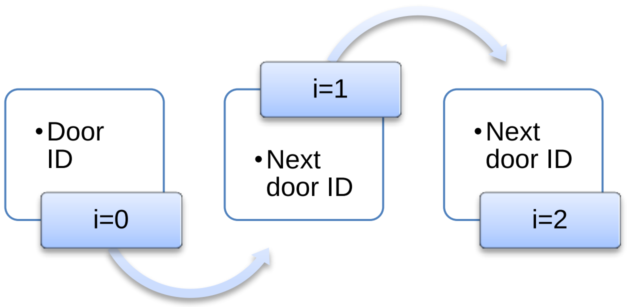


Figure 4-3: Path Array Design

The Figure above shows the outline of PA. PA consists of Lock$_{id}$ stored at each index (i = 0, 1, 2...). PA holds the Lock$_{id}$ that should be accessible by a user in a sequence. Server digitally signs the PA stored inside the Ticket. Index i value starts from 0 and increments each time a user passes a door. This index value points to the Lock$_{id}$ that the user needs to visit. Hence, the contents of the Ticket will be as follows.

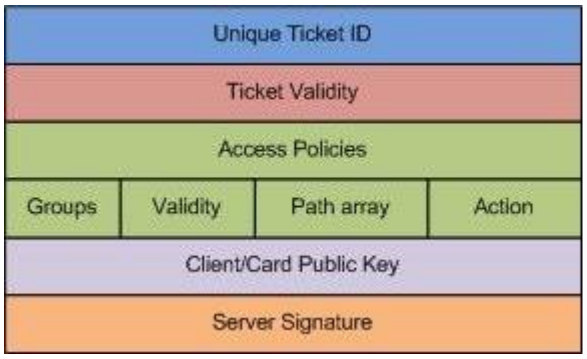


Figure 4-4: Ticket along with Path Array

4.2.6 Smart Card and biometric data

Smart cards, among the others, can be used to store very sensitive data as those of biometric data (e.g images) in that way in which only authorized entities are entitled to get access to them. Biometric data provide high confidence with regard to the user identification and authentication because of their in heritage properties. This means that the complexity to reproduce biometric data that belong to a specific entity is very high. Storing biometric data in smart cards can be used in order to achieve two factor authentication. This is because the biometric data can be stored securely in smart card in such a way that only the holder of the card can gain access to the biometric data.

4.2.7 Face Recognition Smart Card Support

To build trust among different types of nodes on the nSHIELD architecture we can exploit the benefits of smart cards and the cryptographic schemes they implement. In the nSHIELD architecture where decentralized components are interacting not only with each other, but also with centralized ones, there is a need for integrating security and interoperability. In this context, we exploit the advantages of smart cards to enable different types of nodes to provide the following security services:

- Allow the secure key management required for establishing secure channels between different nodes.
- “Anonymous” Authentication e.g., between the sensor and central or other distributed components in the train network.

- Protecting message integrity, for sensor data in the train network among the node and the central system.

In this context, smart cards as mentioned in D3.2 can be used in order to implement an off-line access rights delegation relying on Physical Access Control System [9]. In this use case the smart card will be used to:

- Generate the session key
- Generate the HMAC
- Store the Path Access

Furthermore, in order to increase the confidence which the service has to the users about their claims regarding their identities, biometric data such as users images can be stored to the smart card and validated every time users trying to access a protected resource. For example, when a user tries to access specific doors the nSHIELD node captures an image of the user and compare its biometric data with those are stored in the smart card. This way, there is not a need communicate with the central directory for all the requests. To incorporate this approach to the nSHIELD architecture the following figure depicts a high level approach for integrating smart cards security services in the use case of face recognition.

The entiraction map between the components is described in Figure 3-6.

4.3 Dependable Distributed Computation Framework

The dependable distributed computation framework (codenamed Atta, for compactness) deals with the deployment and execution of applications in a distributed system. Its primary role is therefore to provide a formalized way to specify how a distributed application is composed, and at the same time to offer a runtime that handles the computation associated with said application after deployment.

An application in Atta is written using the dataflow paradigm, expressed as a graph comprising edges and vertices: edges represent data, and vertices represent implementations. Implementations may be just code sections to be executed, or they may represent a sub graph. Therefore we can see that the description of an application may become rather complex. To tackle such complexity and also to simplify abstraction/refinement development approaches, the descriptor of an application is hierarchical. This means that we do not store the entire application graph within one descriptor file, but we rather let descriptors reference other descriptors.

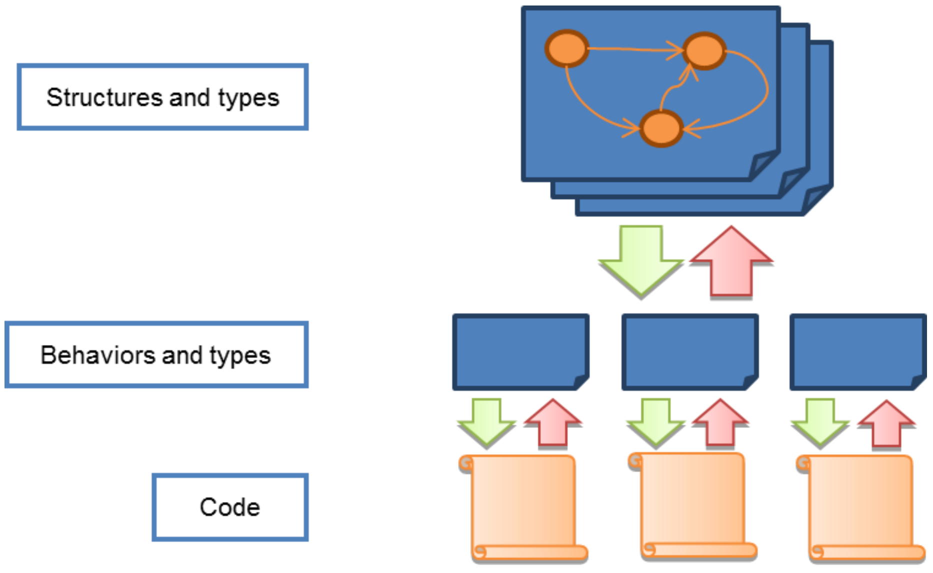


Figure 4-5: Top-down and bottom-up design flows

Figure 4-5 shows how a new application can be designed by using the Atta paradigm under a top-down approach. First, the application is defined as a directed graph. In this phase the semantics of each code section is undefined: only the interactions between code sections are important. Two kinds of *artifacts* are hereby identified: the data *types* for the information transferred between vertices and the blocks (called structural implementations, or simply *structures*) of vertices interconnected by edges. A vertex within a structure may contain either another structure or a code section. The artifacts corresponding to the latter are called behavioural implementations or *behaviours;* the specification of behaviour describes how to build the sources and access the resulting library. Please note that each artifact will have its own descriptor file and such file will be published (along with any source code, in the case of behaviours) in a versioned repository for deployment.

The runtime of the framework is quite complex, due to the fact that it must be distributed and fault tolerant. The application is being run on a set of *worker nodes* (here called “wnodes”), that are embedded nodes capable of accepting workload. Not all worker nodes necessarily perform computation at a given moment, some of them being redundant resources that may be called upon when a computing wnode fails.

In addition to wnodes, other sets of nodes that are relevant within Atta are the following:

- Synchronization nodes (“snodes”): nodes that have the task of synchronizing communication between worker nodes; their primary role is that of triggering an event to all of its listeners as soon as new data is available;
- Persistence nodes (“pnodes”): nodes that persist data; they are nothing more than a (distributed) database that wnodes can use to load/store data;
- Client nodes (“cnodes”): nodes that can inject or extract data for the application by leveraging the pnodes; they represent the external interface to the application, so they also allow composition of distinct applications;
- Directory nodes (“dnodes”): nodes that handle the participation of the other nodes to a specific application, also performing authentication and possibly authorization; they represent the authority within the framework, by grouping nodes and handling security;
- Repository nodes (“rnodes”): nodes that provide (parts of) the application to be deployed on the wnodes; they hold versioned repositories, i.e., Subversion or Git or Mercurial;

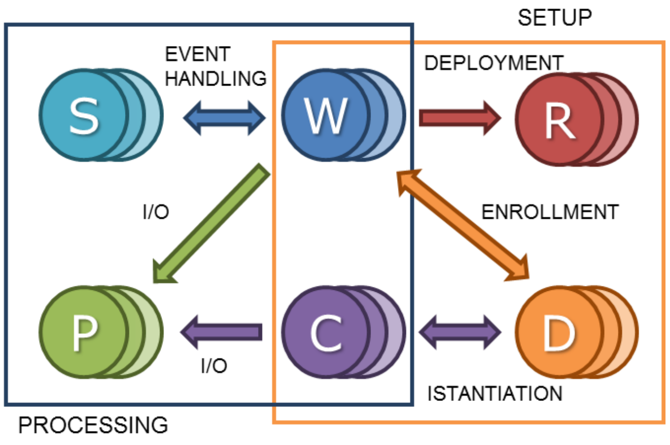


Figure 4-6: Architecture of the Atta runtime

Figure 4-6 shows the corresponding architecture, where each node is marked with a letter denoting its *role* (e.g., C for client). It must be noted here that an embedded node can participate on multiple sets and consequently can take on multiple roles within this software architecture. Clearly, multiple independent devices for each role are envisioned to improve the robustness of both the execution of the code and the consumption of the computed data. Consequently, the roles as defined above must be considered logical components of the runtime, rather than disjoint physical embedded platforms.

The setup phase deals with deployment and preparation for execution, while the processing phase may seem similar to a conventional elaboration on a distributed platform. However, in addition the framework deals with communication and synchronization issues, also providing other useful services such as real-time auditing.

4.3.1 Within the People Identification use case

In the context of the People Identification scenario, the framework covers the need to deploy versioned code in an automatic way, and to avoid single-point-of-failure situations as much as possible. This enhancement to dependability originates from two features:

a) The application can be explicitly designed to be distributed: if multiple cameras for identification are available, we can tolerate temporary downtimes or permanent failures to several cameras without compromising the operation of the system.

b) We can introduce redundancy in some resources: apart from the presence of multiple cameras, there may be other computational nodes related to communication or persistence that are redundant. The middleware is responsible for handling failures and preserving the execution of the application.

Consequently, Atta features are not relevant to a particular scenario: as long as automatic deployment and dependable computation are useful, the framework is valuable. In order to make an application “Atta-compatible”, two steps are required in practice:

1. A redesign of the application using the dataflow application metamodel: this part requires some effort, in order to split an existing application into its independent parts and identify how the latter communication between each other; alternatively, the application can be designed from the start to have a distributed nature.

2. The installation of the runtime on the desired nodes, possibly with some configuration required for authorization.

Clearly, the presence of a middleware requires the nodes to feature sufficient computational capability. In particular, the nodes are required to run an operating system capable of handling the Java runtime. Such nodes will take at least the role of wnodes and will contribute to the execution of the face recognition routines. Additional computational resources (in terms of wnodes, snodes, dnodes and rnodes) can still be supplied by server-class nodes.

5 nSHIELD People identification scenarios

This section defines and describes the nSHIELD scenarios related to the “People identification at the stadium” use case. Three scenarios have been selected:

- people identification at the turnstile,
- people identification on the tribune,
- identification and access management of the stadium personnel.

The final demonstrator will be developed following the identified scenarios.

5.1 Scenario 1: people identification at the turnstile

The first scenario of the use case “People Identification at the Stadium” takes place at the entrance of the stadium, where the identity of people is checked to prevent unauthorized access or to refuse the access to untrusted or unwanted people.

At the entrance of the stadium, at the turnstile, the identity of a person is checked twice using two different technologies: face recognition and smart-card secure services. The identification is based in both cases on the use of a biometric profile that is associated univocally to a person.

The first technology, face recognition, is adopted to dynamically acquire the biometric profile of a person from a video stream. A camera, equipped with the ETHSecuBoard (Prototypes 7 and 37), is installed near the turnstile and the security system asks to the person to position and stand in front of the camera for the face recognition. The face recognition algorithm automatically detects the presence of a face in the video stream, calculates the best image in the stream and extracts the biometric profile. It is required a collaborative approach to the user, in order to avoid situations where the face is obfuscated by clothes or voluntarily altered, trying to hide the real identity. The level of collaboration required in real conditions is very limited and is determined by the ICAO standard: the image selected in the video stream that is used to extract the biometric profile is the image that better satisfies the requirements of the ICAO standard. For security reasons, if during the sampling interval the face recognition algorithm is not capable to identify an image containing a photo of the face that satisfy the ICAO standard, the turnstile doesn’t open. In this situation, the identification system asks to the person to remove any object or cloth obfuscating the face and to reposition in front of the camera. The procedure is repeated five times and, in the case it fails, the identification process is demanded to the security personnel present at the entrance of the stadium.

The second technology, adopts the security services offered by the access rights delegation service developed by TUC (see deliverables D3.1, D3.2 and D3.3, Prototype 6). This solution provides a second source of information for the biometric profile: a smart card is used as an electronic id card that contains the biometric profile of the user. The access rights delegation service guarantees that this information is read from it in a secure way. Furthermore, this service ensures that, after the entrance of the stadium, the access of the user to the correct area of the stadium and the identification of the correct seat is kept passively under control. This functionality helps the user to move in the stadium building, avoiding the possibility to get lost or access areas restricted to the stadium personnel.

With this approach, the identification of a person is based on two independent sources of information that are used to crosscheck the biometric profile acquired by the camera and provided with the smart card. This double check increase the security and the dependability of the identification process, because the two technologies adopted for the acquisition of the biometric profile enforce themselves reciprocally. The weak aspects of the face recognition are enforced by the support offered by the smart card and the possibility that the smart card is not used by the real owner is excluded or confirmed by the face recognition.

After the biometric profile has been acquired from the two sources, the first check consists in comparing the two versions to understand if they correspond. If a first matching is found, the biometric profile is stored in a temporary archive. If the profiles don’t match the access to the stadium is refused and the security personnel take in charge the issue.

The biometric profile stored in the temporary archive is then used to find a matching profile in the central data base containing the biometric profiles of trusted, untrusted and unwanted people. The access to the stadium is granted only in case of matching with the profile of a trusted person.

An alternative solution, which can be adopted i.e. for occasional visitors, is the use of a ticket (and a ticket reader) as a substitute of the smart card. The security and dependability of this solution is lower than the previous one, but it is still acceptable. With this solution, it is possible to ensure that the owner of the ticket is the person that has been identified by the camera.

To increase the dependability of the system and reduce the overall costs, the Dependable Distributed Computation Framework (DDCF, Prototype 14) is used to physically partition and distribute the software components of the people identification system. The face recognition system is composed by three modules: the face finder module (FF), the ICAO module (ICAO) and the face recognition module (FR). The DDCF is capable to execute and manage the FF and ICAO modules directly on the camera, while the FR module is executed and managed on the central server.

The following figure illustrates the identification procedure of a person at the turnstile.

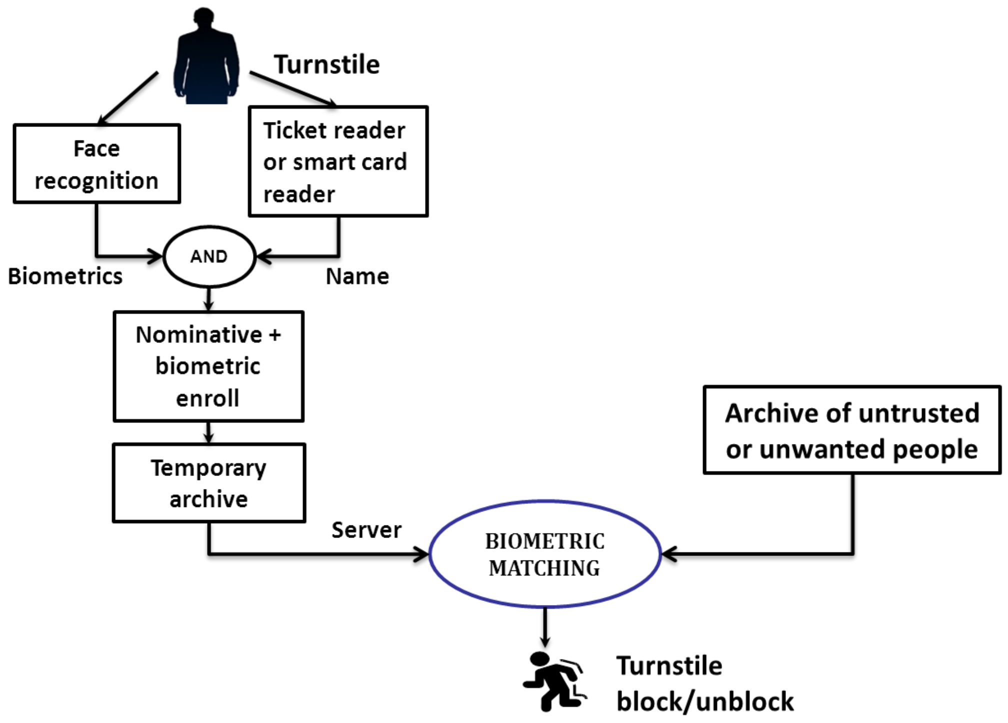


Figure 5-1: Scenario 1 conceptual flow

5.2 Scenario 2: people identification on the tribunes

The second scenario of the use case “People Identification at the Stadium” takes place in the stadium, where the face recognition system is in charge of the dynamical identification of people present on the tribunes. In this scenario the system is used to support the activities of the police and of the security, dynamically identifying people that are responsible for behaviours not allowed on the tribune, clashes between supporters, throwing of objects, damages to the stadium, etc.

The most important advantage of this solution is the possibility to discover the identity of people in real time: currently, the approach adopted for people identification is to analyse the video registered by security cameras and manually find a matching with the criminal database of the police. Unfortunately, this approach has many disadvantages: it can be applied only after the critical situation happens, it relies entirely on the expertise of police personnel, introduces wide time delays in the investigations, etc.

The prototypes involved in this scenario are the Eurotech SecuBoard (Prototype 7) and the Dependable Distributed Computation Framework (DDCF, Prototype 14).

The people identification system is based on a set of cameras, based on the ETH SecuBoard, that are capable to execute a distributed version of the face recognition software, which is managed by the Dependable Distributed Computation Framework.

The cameras are equipped with appropriate optics and are located around the playing area. The positions are selected considering the technical capabilities of the optics and trying to cover, with their field of view, the area of the tribunes. The selection of the positions is strongly influenced by optical issues (see the following figures):

- the field of view of the cameras is limited;
- the area of the tribune to be scanned by the camera is very wide;
- the structure of the tribune implies that the identification is performed on different focal planes;
- for dependability reasons, every area of the tribunes must be covered by two cameras.

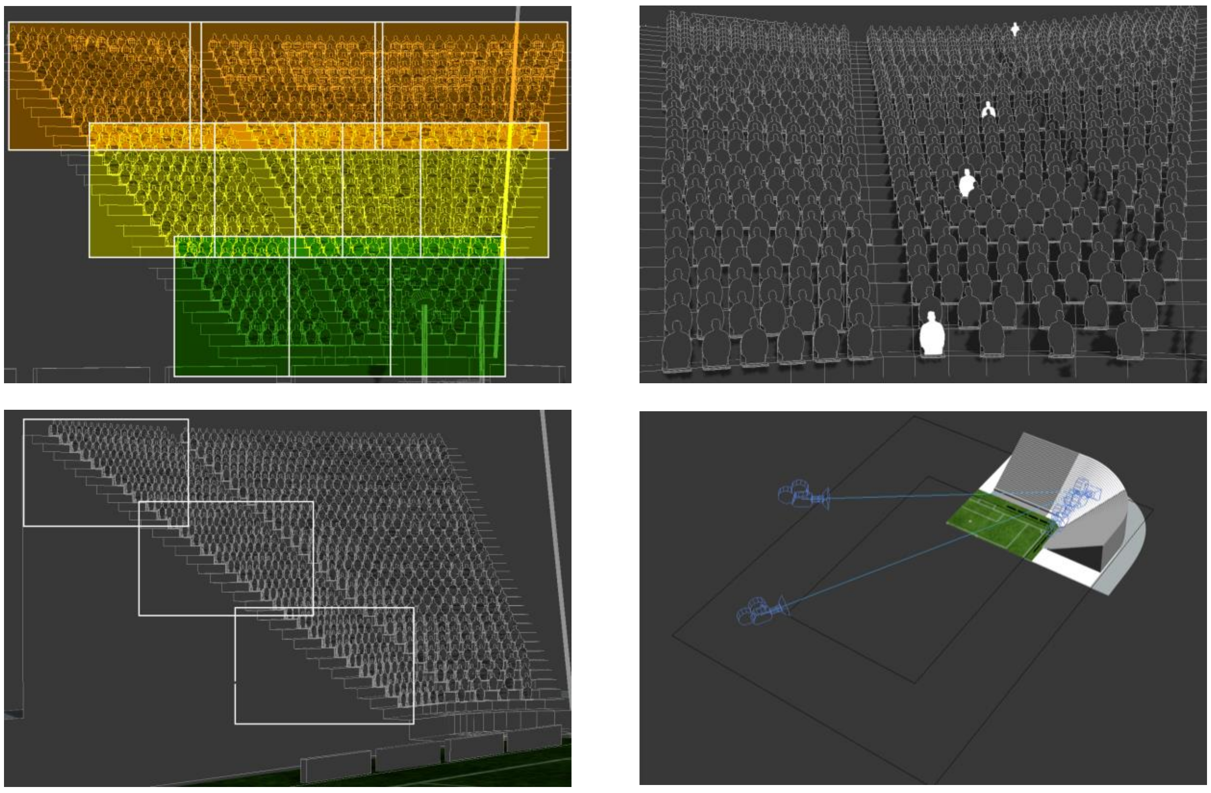


Figure 5-2: Examples of camera position, focal planes and areas coverage

Each camera is equipped with the ETH Secure Board, on which run the Face Finder module and the ICAO module of the face recognition system. The Face Recognition module is hosted on the central server, where it is execute in several instances. The Dependable Distributed Computation Framework is responsible for the software management of the people identification system:

- it is responsible for the execution of the FF and the ICAO modules,
- it is responsible for the execution of the instances of the FR module,
- it manages the status and the execution of all the modules,

- it is in charge of the management of the communications between the modules.

The security cameras positioned around the playing area are fixed installation but this is not a strict requirement. The ETH SecuBoard is an nSHIELD micro node that means an embedded system with low power consumptions, small dimensions, and battery powered and completely autonomous. These features allow the possibility to develop a rugged and mobile version of the previous camera that can be positioned in the stadium depending on the actual situation and necessities. A typical example of the use of a mobile camera is when some areas of the tribunes are covered by the smog caused by smoke bombs: in this case the view angle of fixed cameras could be obfuscated by the smog, while a mobile camera can be positioned by the security conveniently, in order to have a clear angle of view of the area. The only limitation of this solution is represented by the optics that, considering the dimension and weight, is not perfectly suitable for a transportable embedded system.

The following figure illustrates the identification procedure on the tribunes.

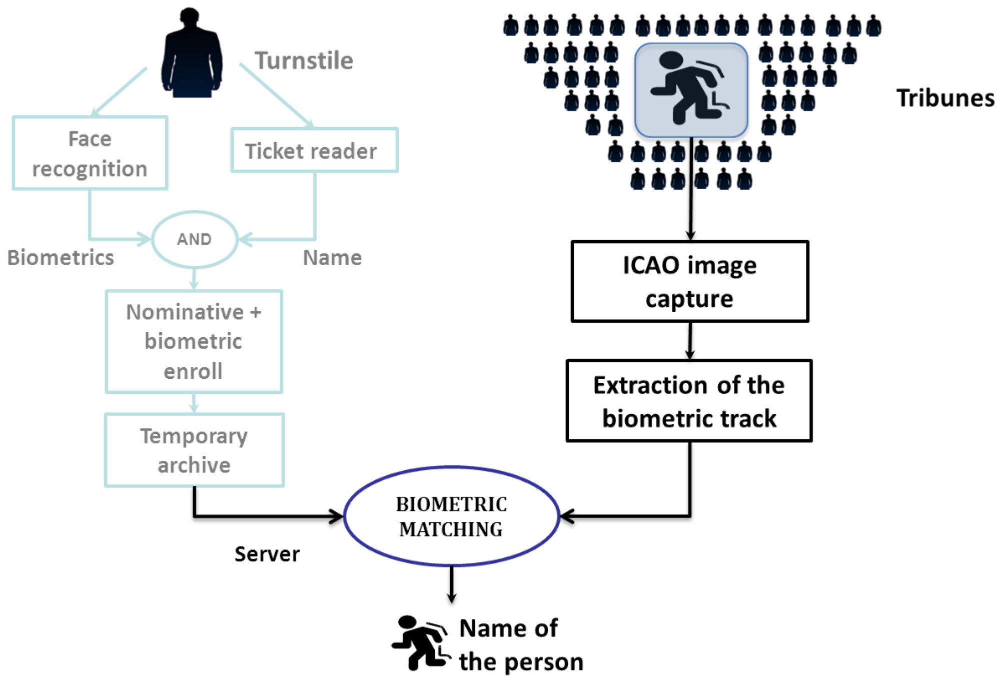


Figure 5-3: Scenario 2 conceptual flow

Each camera around the playing area physically scans the assigned area of the tribunes and tries to identify faces. Considering the number of people present on the tribunes, this simple task becomes a heavy activity from a computational point of view. For this reason, the decision to partition the face recognition application in modules and to adopt the DDCF to execute and manage the modules is strategic in this scenario. The face recognition application doesn't run entirely on the ETH SecuBoard and the computational power can be exclusively reserved to the FF module.

At this stage, the face recognition procedure can be completely autonomous, identifying a set of faces instead of a single face as in scenario 1. The number of faces identified in an image of the video stream is in any case low because, to guarantee high identification accuracy, the area covered during the scan must be limited to 10-20 squared meters. The camera must have a field of view not to large, in which the details of the people faces are well focalized.

The remaining steps of the people identification are the same as in scenario 1, with the exception that both the ICAO score computation and the extraction of the biometric profiles are executed on a central computation unit (i.e. and industrial PC). In this case the profile matching is performed between the biometric profiles acquired by the cameras and the biometric profiles of the people that have been registered at the turnstile of the stadium.

5.3 Scenario 3: identification and access management of the stadium personnel

The stadium personnel require access to various rooms before, during and after a game for management purposes. Since these rooms might be marked by the stadium management team with different security levels (e.g, minimum, medium, high), smart cards can be exploited to support the required security levels without needed communication with the centralized system. The latter, will be established only in the cases where an unauthorized access is attempted. The following figure illustrates the case of requesting access to a room classified with a “high” security label. In that case, we assume that the high level can be achieved through biometric data validation. On the other hand, when the room label is “medium” the corresponding trust can be achieved via the procedure described in Section 4.2.5.

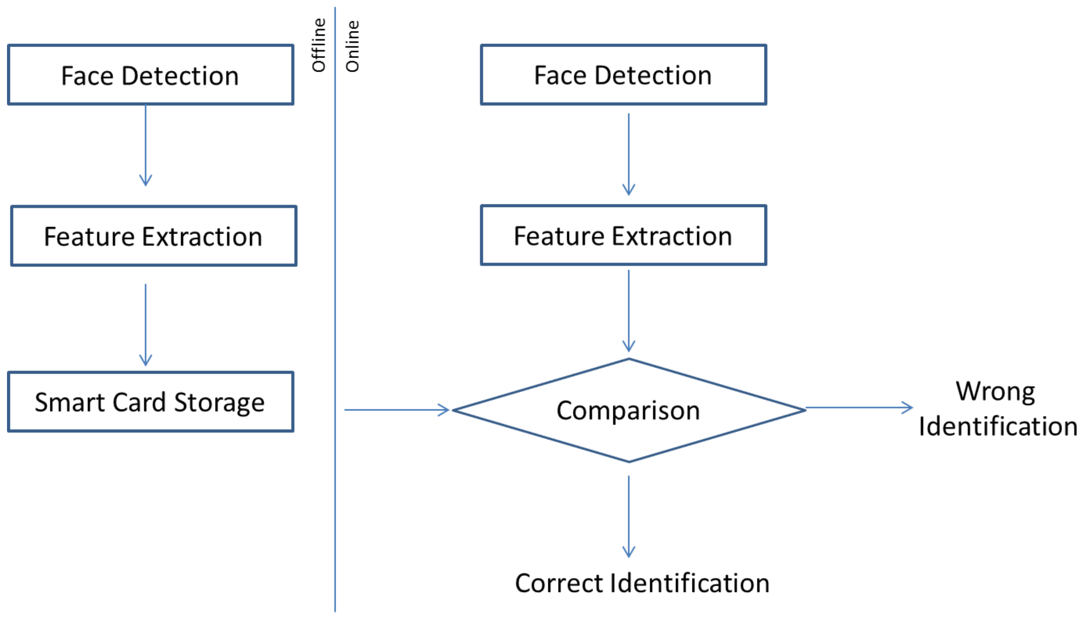


Figure 5-4: Scenario 3 conceptual flow

6 People identification system demonstrator integration

This section describes the approach to system integration planned for the “People Identification at the Stadium” use case.

6.1 Face recognition for people identification

The face recognition system can be integrated in the scenarios described in the previous sections through an API provided by the various software modules.

As already described, the face recognition software is based on three modules:

- the face finder module (FF),
- the ICAO module (ICAO) and
- the face recognition module (FR).

These components cooperate to implement the recognition and identification procedure illustrated in the following figure:

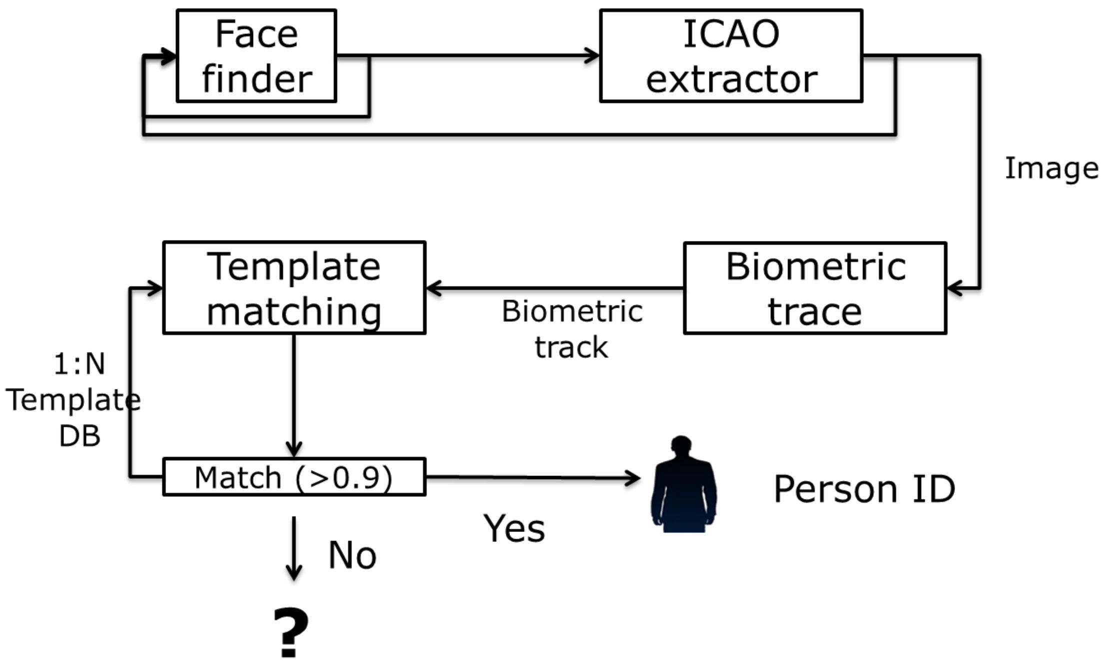


Figure 6-1: Face recognition and identification procedure

The face recognition application can operate in two different modes: “Enrol” mode and “Transit” mode. The enrol mode is used to populate the data base with the biometric profiles of the people that will be accepted by the system. The transit mode is used to dynamically recognize and identify the people that pass (“transit”) in front of the camera. The following diagrams illustrate the operations performed in this two working modes:

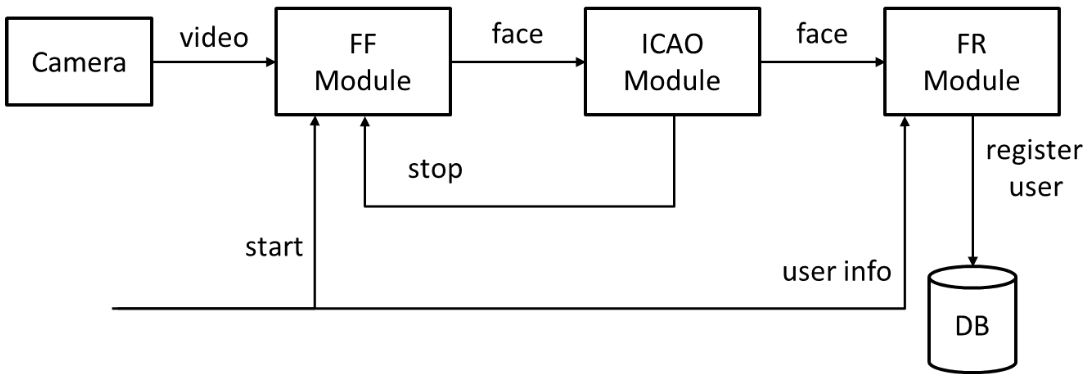


Figure 6-2: The enrol mode

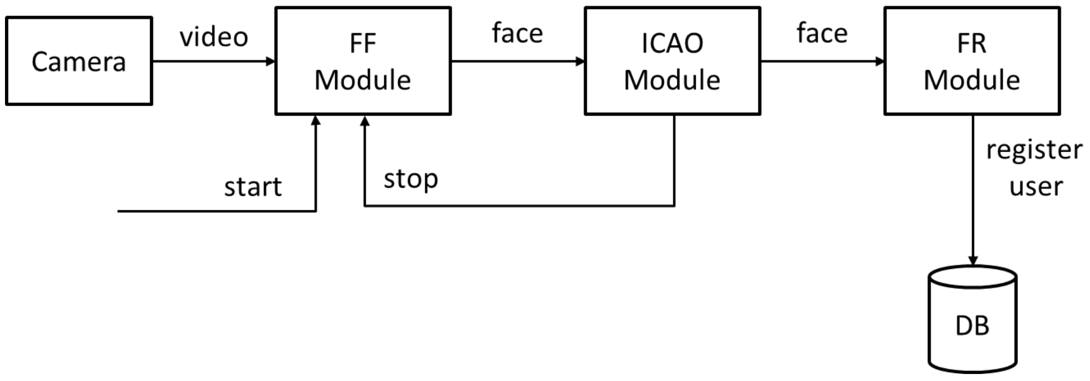


Figure 6-3: The transit mode

The API exposed by the modules will be based on standard RPC.

All the face recognition modules will configurable using a simple configuration file or through a web interface.

6.1.1 Face Finder Module (FF)

The face finder module is responsible for the acquisition of the video stream, for the analysis of the video stream itself and for the generation of the output messages when a human face is found in the input stream. It will provide the following RPC functions: start and stop of face detection, keep alive with feedback on the module status.

Every time a face is detected in the video stream, the extracted features are passed to the ICAO module that accepts them as input data and elaborates them with an appropriate function.

6.1.2 The ICAO module (ICAO)

The ICAO module is responsible for the selection of the best image in the set of images identified by the FF module. The module will provide the following RPC functions: ICAO scores computation and keep alive with feedback on the module status.

Once the module identifies a face with an ICAO score that allows the identification, the “stop” function of the FF module is called and the best result of this detection phase is sent to the face recognition module.

6.1.3 The face recognition module (FR)

The face recognition module is responsible for the extraction of the biometric profile and for the identification of a matching profile in the data base. This module can run in two different modes: “Enrol” and “Transit”. In “Enrol” mode the detected biometric profile and personal information of a person are stored in the data base. In “Transit” mode the module search the data base for the biometric profile matching the one extracted from the video stream. The module will provide the following RPC functions: extraction of the biometric profile, user identification and keep alive with feedback on the module status.

6.1.4 Smart card manager

This component is responsible to read the encrypted biometric profile of a person from his/her smart card. It works with the access rights delegation module to setup a secure session during which the biometric profile is collected from the smart card using a common smart card reader. The module will provide the following RPC functions: open and close a secure session, get the session HMAC and get the biometric profile stored in the smart card.

6.2 Smart Card and Access Delegation

The smartcard communication structure has been introduced in Section 4.2. The general structure of a command in smartcards is illustrated in the table below.

Table 6-1: Smartcard request command format

Header				Data	
CLA	***INS***	***P1***	***P2***	***Length***	
Class where the command lies	*The command itself*	*Command first parameter*	*Command second parameter*	Data Length	Additional Data

The command can be issued towards the smartcard using the underlying communication of the terminal and the smartcard terminal (e.g. serial communication).

For every command issued toward to the smartcard there is a response which its format illustrated in the following table.

Table 6-2: Smart card response command format

Data	***Response Status***
The data returned by the smartcard	*Show the result of the requested command ,whether the command is successful or failed, and the reason of failure*

6.2.1 Smartcard File System and Data “Storage”

Smartcards file system structure is similar to those used in operating system. Particularly the ISO-7816 part 4 defines the structure of the file system as illustrated in the following figure. The master file (MF) can be considered as the root directory, while the dedicated and elementary files are the directories and the data file, in UNIX like operating system, correspondingly.

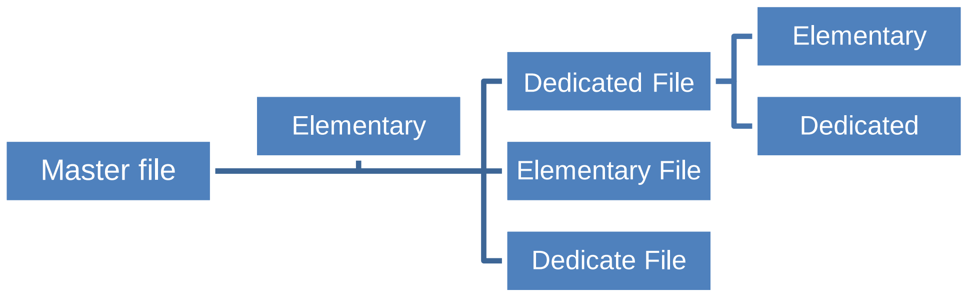


Figure 6-4: The logical structure of file system in Smartcards

In smartcards different kind of data can be stored either dynamically or statically, though their capacity is limited. For example, users' data or cryptographic keys for secure transactions can be stored. The header in data files defines also the access control rights. Every directory creates a security domain inheriting the security policy of its parent. The files in the smartcard can be protected with multiple ways:

- Different PIN
- Message authentication code
- Access control restrictions (read, write permissions)
- Digital signatures

This depends on the features incorporated in the smartcard.

Building secure communication

In the scheme proposed in section 4.2.4, in order to issue a smart card, the related component (e.g. micro node) should create a request for issuing a smart card. This request will include the serial number of the component and will be forwarded to the central authority. If needed, depending on the type of service, the central authority will check the register status of the requested component and afterwards will generate a new smart card. In the new smart card will be installed the following information:

- Node's serial number.
- Node's secret key.
- Node's id.
- Node's auth key.

The generation of secret keys will be based on the following types:

```
Encryption-Key = AES-256 (Central Mother Key XOR Node's Serial Number)
Auth-Key = AES-256 (Central Mother Key XOR Node's ID)
```

We should note that in this scheme we assume that the central authority has also a TPM for generating the secret keys in a secure way for the issued tokens, while all the smart cards are issued by a (trusted) central authority. The generated keys will be unique since they are related with node's serial number, which is unique.

The node, as a smart card is issued can exploit its security feature for providing confidentiality, integrity or/and authenticity services. The provided services depend on the application. For instance, if there is a requirement to provide confidentiality services to the data sent to the central authority the following procedure will be take place:

1. The node will send the data to the smart card.
2. The smart card encrypts the provided data, using the secret-key installed into the smart card during the registration, and forwards them to the node.

3. The node sends to the central authority the encrypted data and its serial number.
4. The central authority generates in the TPM the corresponding secret key using the serial number sent by the node. Note that the key is not “extracted” from the TPM.
5. The TPM decrypts the data and send and acknowledgement to the node.

A very similar approach will be followed when an authentication is needed. Particularly:

1. The node will send to the smart card a random number that will be used as the data require validation.
2. The smart card using the Auth-Key and the random data generates the MAC and forwards it to the node.
3. The node sends to the central authority the MAC including the random number and its id.
4. The central authority generates in the TPM the corresponding auth key using the id.
5. The central authority using the auth-key produces a new MAC and compares it with the one received by the node. If those two MACs are matched the central authority sends to the node a successful response otherwise a failure occurs.

These procedures can be combined in order to provide confidentiality and authenticity services simultaneously, depending on the requirements.

6.2.1.1 Using smartcards for security services: authentication example in the context of nSHIELD

For instance, consider the case where the overlay should authenticate a Micro-Node that incorporates a smartcard module. In that case the overlay generates a challenge and sends it to the micro node. The Micro-Node passes the challenge to the smartcard and requests it to create a message authentication code (MAC), assuming that we rely on symmetric key cryptography. The smartcard generates the MAC and sends it back to the Micro-Node that forwards the result to the overlay. The overlay can validate the received MAC either using a TPM or a software based security service. This procedure is illustrated in high level in the figure below. Note that the symmetric keys required by the Micro-Node can be either pre-installed in the smartcard or be generated dynamically in the smartcard itself.

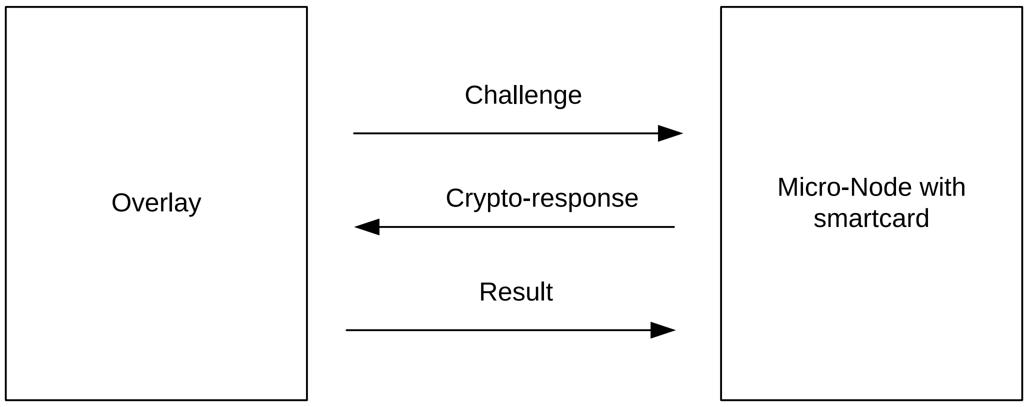


Figure 6-5: Example of authentication using smartcards. The overlay authenticates a Micro-Node

A similar procedure can be followed in the case where Micro-Node needs to authenticate the overlay. Particularly, the Micro-Node requests the smartcard to generate a random number which is forwarded to the overlay. The overlay generates the corresponding MAC and sends it back to the Micro-Node which requests the smart card to validate the generated MAC. Depending on the result it creates either success or failure response that is sent to the Micro-Node. Note that the smart card may not be able to validate itself the MAC. In that case, the smartcard will generate the MAC using the same challenge and the final validation will be accomplished by the Micro-Node by comparing the MACs received by the overlay and the smartcard. This procedure is depicted in the figure below.

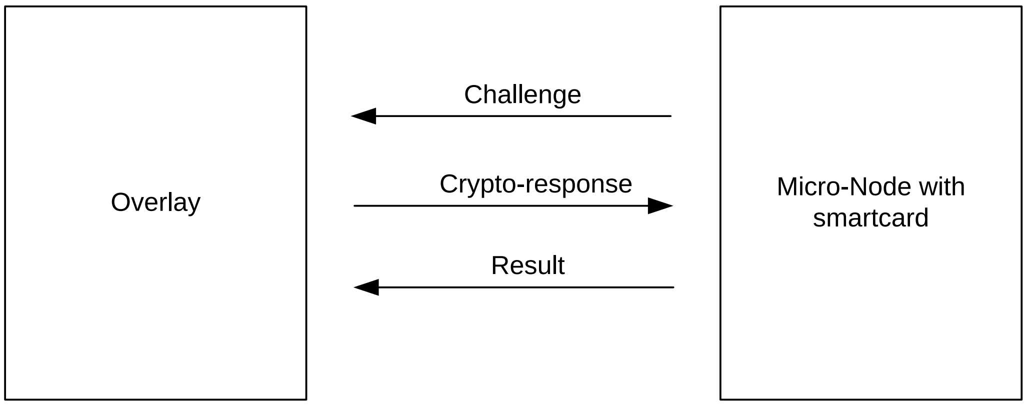


Figure 6-6: Example of authentication using smartcards.
The Micro-Node authenticates the overlay.

6.2.2 Access rights delegation

The mechanism of the approach for access rights delegation illustrated in section 4.2.5, can be described as follows.

After the creation of trust between the entities of offline PACS [9], `Lock` now processes the contents of PA, and then checks for its own ID at the current index i, if it is found then `Lock` performs three steps as follows,

- Increment index i
- Generate HMAC and write it to `Card`
- Grant access to the user

If the $\mathtt{Lock}_{id}$ present at index i does not correspond to `Lock` own id, it then it denies the access and logs the user action.

6.2.2.1 Incrementing Index i

The path array PA contains lock ids stored inside it. Only the relative matching `Lock` is allowed to increment i value by one. At the time of generation of `Ticket` by the `Server`, it also generates a HMAC to be used by the first lock in the PA. The `Lock` located at the first index of PA makes use of this HMAC to ensure that no illegal modifications are done on the smart card. The Index of PA starts from the value 0. For instance, consider the below path array. This path array consists of lock ids B, A and C which should be followed in that order by the user.

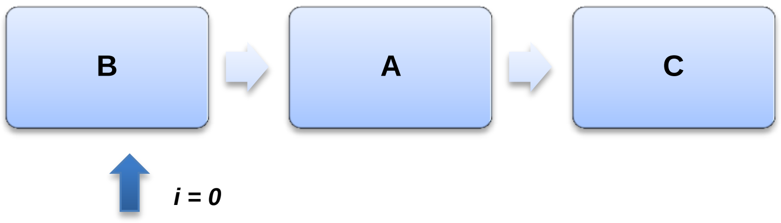


Figure 6-7: Ticket Incrementing the index value

In the figure above, the current value is 0 and PA[0]=B. Only the lock with id ‘B’ can increment the i value further.

6.2.2.2 Generating Hash using HMAC

Lock creates the HMAC after incrementing the i value. HMAC stands for Hash Based Message Authentic Code. It calculates the message authentic code using a cryptographic hash function and shared secret key. In the offline PACS scenario, geographically dispersed locks securely exchange the messages among them by using message digest. HMAC is necessary in offline PACS scenario to ensure the integrity of smart card contents. The process of creating HMAC is as shown in the formula below.

$$HMAC(K,m) = H((K \oplus opad) \parallel H(K \oplus ipad) \parallel m)$$

where:

- K is the shared secret key
- m is the message to be protected
- opad is outer padding (0x5c5c….)
- ipad is inner padding (0x3636….)
- H is the cryptographic hash function (MD5, SHA etc.)
- || is concatenation
- ⊕ is the exclusive-OR operation

The locks inside the facility were pre-installed with $\mathtt{key}_{\mathtt{shared}}$. Concatenating the $\mathtt{key}_{\mathtt{shared}}$ with $\mathtt{Lock}_{\mathtt{id}}$ generates the secret key $\mathtt{key}_{\mathtt{secret}}$. Message $\mathtt{m}$ in this context indicates the current index i value, and the rest of them use default parameters.

While hash generation, $\mathtt{key}_{\mathtt{secret}} = \mathtt{key}_{\mathtt{shared}} \mid\mid \mathtt{Lock}_{\mathtt{id}}$.

Generation of HMAC is as following:

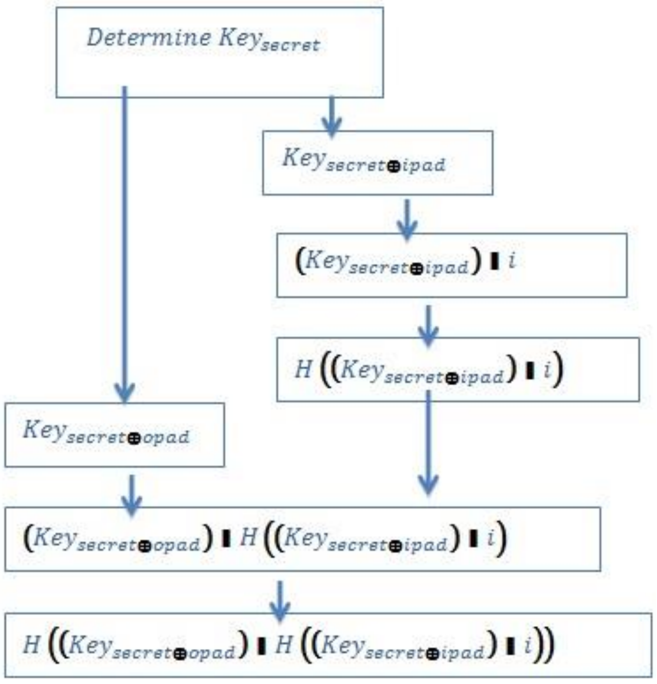

Determine Key_{secret}
$Key_{secret \oplus ipad}$
$(Key_{secret \oplus ipad}) \parallel i$
$H((Key_{secret \oplus ipad}) \parallel i)$
$Key_{secret \oplus opad}$
$(Key_{secret \oplus opad}) \parallel H((Key_{secret \oplus ipad}) \parallel i)$
$H((Key_{secret \oplus opad}) \parallel H((Key_{secret \oplus ipad}) \parallel i))$

Figure 6-8: Process of HMAC creation

In the above figure, H can be any hash algorithm.

6.2.2.3 Generating Hash by the Locks in the Sequence

The overall concept of the artefact is to enforce the user through the path specified by the administrator. The user should attend this path in sequence. Hence, if one lock grants access to the user, which is present at index zero of PA i.e., PA[0], then the next lock in the sequence, which is available at PA[1] should be able to prove that the user has already passed through the door mentioned at PA[0].Verification of the hash value generated by the lock present at PA[0] solves the above issue.

When the user presents his smart card at a `Lock` mentioned in the PA, the lock allows access only if it has confirmed that the user has already been allowed access by an earlier lock. During the verification process of HMAC `Lock` always uses the previous index for key generation and while in a hash generation process it uses its own $\mathtt{Lock}_{\mathtt{id}}$. Current lock initially checks i value.

Scenario 1: If i=0

Then the lock knows that it is the first lock in the order. It then checks whether the value present at PA[0] matches its own lock id. If the id is not equal to its own id, it will log the user activity and deny the access.

Lock has the authority to perform further actions if the value at PA[0] matches its own id. It then verifies the HMAC stored by the `Server` on the `Card`, to make sure that nothing has been changed illegally. It will then increment i value by one and will generate HMAC by using the secret key $\mathtt{key}_{\mathtt{secret}}$. In this scenario, $\mathtt{key}_{\mathtt{secret}}$ results from concatenating the $\mathtt{key}_{\mathtt{shared}}$ with its own $\mathtt{Lock}_{\mathtt{id}}$.

Scenario 2: If i>0

If i value is greater than zero, then lock confirms that the user has already accessed some doors in the sequence. Hence, it will confirm its own authority that it can change the contents of the card by looking up for its own lock id, and then generates hash to verify the hash stored by the earlier lock. Now, the `Lock` increments i value by one and generate a new hash to be used by the next lock in the series.

Verification steps by current lock in action are as follows,

- Step 1: reads current i value
- Step 2: Looks up present at PA[i]
- Step 3: If the value of own =PA[i], then proceed to step 4 else go to step 10
- Step 4: Verify the HMAC hash stored on smart card (generated by previous lock)
- Step 5: If the hash can be verified, continue else go to step 10
- Step 6: Increment i value by one
- Step 7: Generate new HMAC
- Step 8: Replace the old HMAC with generated HMAC to be used by next lock
- Step 9: Allow access and stop
- Step 10: Deny access and log user activity onto the card

Using above procedure the n^{th} lock will verify that the user has accessed the $(n-1)^{th}$ lock, and this process continues with all the locks.

6.3 Dependable Distributed Computation Framework

The role of the Atta framework in this scenario is to support the deployment and execution of the people identification algorithms.

As such, the integration of the framework is mostly a design concern: an application must be (re)designed using the specific dataflow model used by the framework. Summarily, each vertex of the application model contains code, while each edge represents a data transfer. The effort from the designer is therefore to think in terms of independent sections of code that interact with each other.

It must be noted that Atta does not force the designer to abandon his compilation and testing flows of choice: as soon as an application model has been designed, each code section can be built and tested in isolation using the preferred tools. With little adaptation, all sections can be linked together to test the whole application in a monolithic way. Consequently, while “Atta-compliance” introduces additional concerns, it still accommodates as much as possible the existing design conventions.

As already discussed in Section 4, there exist three kinds of artifacts known to the framework: types, behaviours and structures. To integrate Atta with the algorithms for the scenario becomes a three-step operation. Referring to Figure 4-5, if a bottom-up approach is envisioned, we have to:

1. Prepare the library objects, namely:
    a. Separate the monolithic code into independent sections;
    b. Wrap each section to provide a proper access function;
    c. Provide an automatic build procedure.
2. Provide a behaviour descriptor file for each library object, in particular we need to specify:
    a. The location of the build script;
    b. Any library dependencies required for execution;
    c. The data types exchanged by the access function.
3. Provide one or more structure descriptor file to connect the behaviours into a full-fledged application; additional types may be identified in this phase.

Descriptor files and library objects must then be published into versioned repositories, in order to be deployed for execution.

Summarizing, the Atta prototype incorporates routines in the form of libraries with supporting descriptor files. The actual interface between the routines and the framework is made through well-defined access functions. The framework then becomes responsible for spawning processes that employ these libraries, and for data communication and synchronization between the nodes running such processes.

7 People identification system demonstrator validation and verification

This section describes the approach planned for “People Identification at the Stadium” use case validation and verification. It will illustrate the validation and verification aspects relevant to single components and the overall system. Finally it defines a list of tests foreseen to be performed.

7.1 Face recognition for people identification

The evaluation of the quality and reliability of the recognition process is fundamental in order to guarantee the required levels of security and dependability. The evaluation can be considered conceptually as a part of the recognition algorithm itself, providing feedbacks that close the retroaction loop. This phase allows the system to adjust the parameters of the recognition algorithm, in order to maintain the maximum level of identification reliability. The problem addressed during the evaluation phase is very complex and plays an important role since from the begging of the system configuration and deployment: during the initial training phase, a controlled biometric dataset is used to train the recognition algorithm and obtain the configuration that guarantee the minimum level of identification reliability. In this case, the evaluation process is used to understand when the system is ready for the deployment (as shown in the figure below).

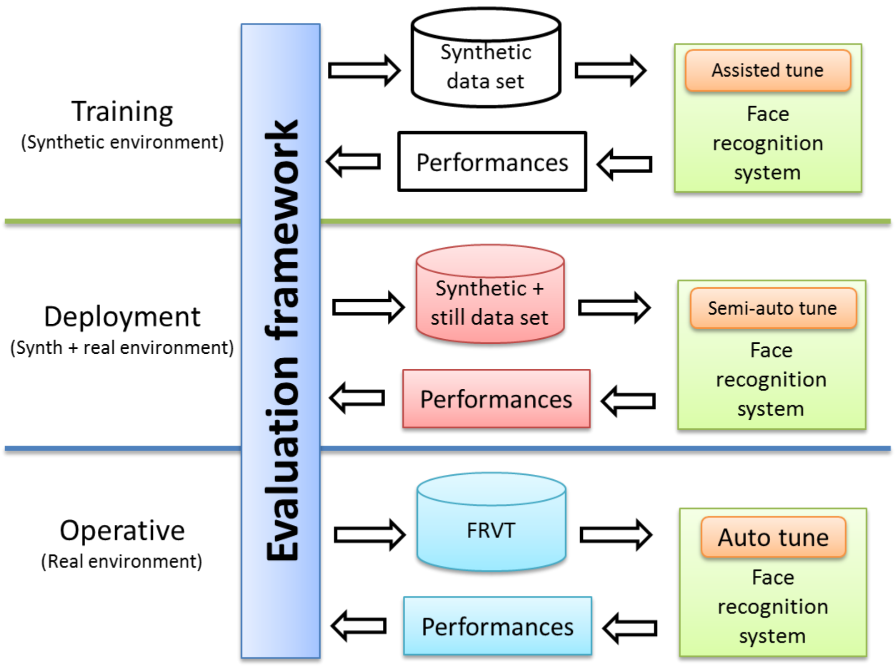


Figure 7-1: The evaluation framework in the recognition system life cycle

From a technical point of view, considering the complexity of the addressed problem, the evaluation process cannot be performed simply by the recognition algorithm but requires a complete evaluation framework that integrates with the recognition system. The evaluation framework is composed by:

- A standard evaluation data set: the Embedded Face Recognition System (EFRS) proposed in nSHIELD adopts a data corpus that must contain at least 50,000 recordings divided into training and validation partitions. The data corpus is constituted by high resolution still images, taken under controlled lighting conditions and with unstructured illumination, 3D scans and contemporaneously still images collected in a real environment.
- A challenging problem that allows the evaluation of the improvement in terms of performance: the identification of a challenging recognition scenario ensures that the evaluation is performed on

sufficiently reasonable, complex and large problems and that the results obtained are valuable, in particular when compared between different configurations and recognition algorithms. The challenging problem identified to evaluate the EFRS consists of six experiments. The experiments measure the performance on still images taken with controlled lighting and background, uncontrolled lighting and background, 3D imagery, multi-still imagery, and between 3D and still images.

- A software evaluation infrastructure: it supports an objective comparison among different recognition configuration and algorithms. The infrastructure ensures that results from different algorithms are computed on the same data sets and that performance scores are generated with the same protocol. To measure the improvements introduced by the EFRS and perform the run time evaluation required by the recognition system we selected the Face Recognition Vendor Test of year 2002 (FRVT) [8] that “provides independent government evaluations of commercially available and prototype face recognition technologies. These evaluations are designed to provide U.S. Government and law enforcement agencies with information to assist them in determining where and how facial recognition technology can best be deployed. In addition, FRVT results help identify future research directions for the face recognition community”. FRVT 2002 consists of two tests: the High Computational Intensity (HCInt) Test and the Medium Computational Intensity (MCInt) Test. Both tests require the systems to be full automatic, and manual intervention is not allowed.

7.1.1 Design of Data Set and Challenge Problem

The design of the EFRS starts from the performance measured using FRVT. It establishes a performance goal that is an order of magnitude greater than FRVT measured performance. Starting from this goal, it introduces a data corpus and a challenge problem that are significant and valuable for real application. The evaluation process, using this data corpus and challenge problem, aims at understanding if the EFRS has reached the FRVT goal and is capable to maintain at runtime this performance improvement.

The starting point for measuring the improvement of performance is the high computational intensity test (HCInt) of the FRVT. The images in the HCInt corpus are taken indoors under controlled illumination. The performance point selected as the reference is a verification rate of 80% (error rate of 20%) at a false accept rate (FAR) of 0.1%. This is the performance level of the top three FRVT 2002 participants. An order of magnitude improvement in performance that we expect from EFRS requires a verification rate of 98% (2% error rate) at the same fixed FAR of 0.1%.

A challenge for designing the EFRS is collecting sufficient data to measure an error rate of 2%. The verification performance is characterized by two statistics: verification rate and false acceptance rate. The false acceptance rate is computed from comparisons between faces of different people. These comparisons are called non-matches. In most experiments, there are sufficient non-match scores because the number of non-match scores is usually quadratic in the size of the data set. The verification rate is computed from comparisons between two facial images of the same person. These comparisons are called match scores. Because the number of match scores is linear in the data set size, generating a sufficient number of matches can be difficult.

For a verification rate of 98%, the expected verification error rate is one in every 50 match scores. To be able to perform advanced statistical analysis, 50,000 match scores are required. From 50,000 match scores, the expected number of verification errors is 1,000 (at the EFRS performance goal).

The challenge is to design a data collection protocol that yields 50,000 match scores. We accomplished this by collecting images for a medium number of people with a medium number of replicates. The proposed EFRS data collection is based on the acquisition of images of 200 subjects once a week for a year, which generates approximately 50,000 match scores.

The design, development, tuning and evaluation of the face recognition algorithms require three data partitions: training, validation, and testing. The EFRS challenge problem provides training and validation partitions to developers. A separate testing partition is being collected and sequestered for an independent evaluation.

The representation, feature selection, and classifier training is conducted on the training partition. For example, in PCA-based (Principle Component Analysis) and LDA-based (Linear Discriminant Analysis) face recognition, the subspace representation is learned from the training set. In vector machine (SVM) based face recognition algorithms, the SVM classifier is trained on the data in the training partition.

The challenge problem experiments must be constructed from data in the validation partition. During algorithm development, repeated runs are made on the challenge problems. This allows developers to assess the best approaches and tune their algorithms. Repeated runs produce algorithms that are tuned to the validation partition. An algorithm that is not designed properly will not generalize to another data set.

To obtain an objective measure of performance it is necessary that the results are computed on a separate test data set. The test partition measures how well an approach generalizes to another data set. By sequestering the data in test partition, participants cannot tune their algorithm or system to the test data. This allows for an unbiased assessment of algorithm and system performance.

The EFRS experimental protocol is based on the FRVT 2002 testing protocols. For an experiment, the input to an algorithm is two sets of images: target and query sets. Images in the target set represent facial images known to the system. Images in the query set represent unknown images presented to the system for recognition and identification. The output from an algorithm is a similarity matrix, in which each element is a similarity score that measures the degree of similarity between two facial images. The similarity matrix is comprised of the similarity scores between all pairs of images in the target and query matrices. Verification scores are computed from the similarity matrix (see next figure).

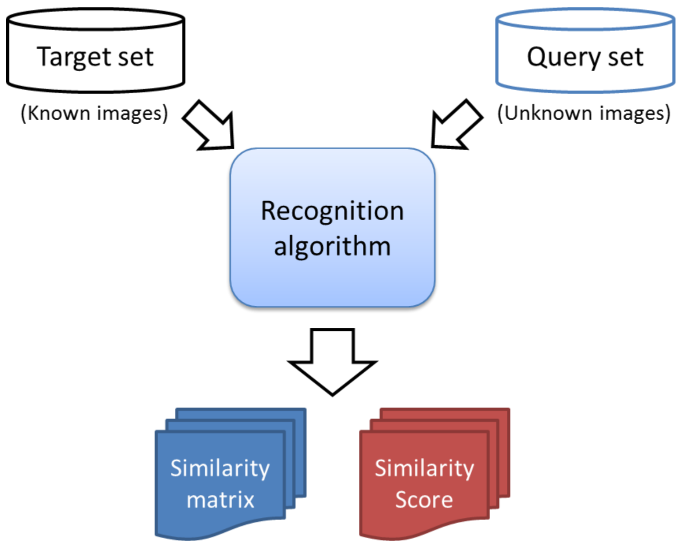


Figure 7-2: The testing protocol.

7.1.2 Description of the Data Set

The EFRS data corpus is part of an on-going multi-modal biometric data collection.

A subject session is the set of all images of a person taken each time a person's biometric data is collected. The EFRS data for a subject session consists of four controlled still images, two uncontrolled still images, and one three-dimensional image. The figure below shows a set of images for one subject session. The controlled images are taken in a studio setting, are full frontal facial images taken under two lighting conditions (two or three studio lights) and with two facial expressions (smiling and neutral). The uncontrolled images were taken in varying illumination conditions; e.g., hallways, atria, or outdoors. Each set of uncontrolled images contains two expressions, smiling and neutral. The 3D images are taken under controlled illumination conditions appropriate for the sensor (structured light sensor that takes a 640 by 480 range sampling and a registered colour image), not the same as the conditions for the controlled still images. In the FRP, 3D images consist of both range and texture channels. The sensor acquires the texture channel just after the acquisition of the shape channel. This can result in subject motion that can cause poor registration between the texture and shape channels. The still images are taken with a 4 Megapixel camera.

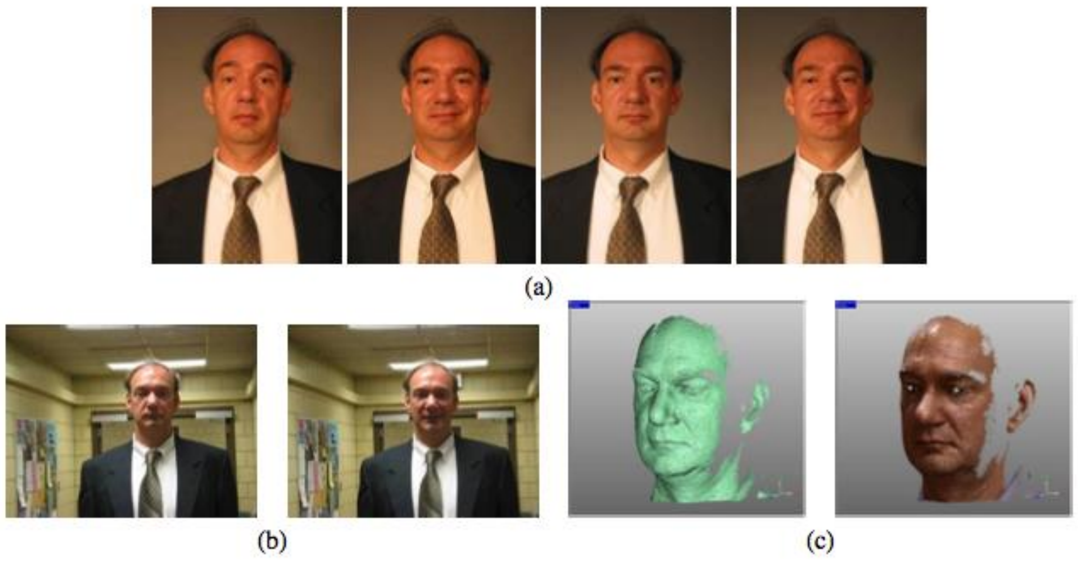


Figure 7-3: Images from one subject session.
(a) Four controlled stills, (b) two uncontrolled stills, and (c) 3D shape channel and texture channel pasted on 3D shape channel.

Images are either 1704x2272 pixels or 1200x1600 pixels, in JPEG format and storage sizes range from 1.2 Mbytes to 3.1 Mbytes. Subjects have been photographed approximately 1.5 meters from the sensor.

Table 7-1: Size of faces in the validation set imagery broken out by category. Size is measured in pixels between the centres of the eyes. The table reports mean, median, and standard deviation

	Mean	Median	Standard Deviation
Controlled	261	260	19
Uncontrolled	144	143	14
3D	160	162	15

The table above summarizes the size of the images for the uncontrolled, controlled, and 3D image categories. The average distance between the centres of the eyes in the FRVT 2002 database is 68 pixels with a standard deviation of 8.7 pixels: the data set adopted for the EFRS satisfies the FRVT 2002 and provide a quality largely better than the FRVT 2002. This means that we follow this standard but we use a source of information that makes the test at least 4 to 6 times harder.

The data required for the experiments on the EFRS are divided into training and validation partitions. From the training partition, two training sets are distributed. The first is the large still training set, which is designed for training still face recognition algorithms. The large still training set consists of 12,776 images from 222 subjects, with 6,388 controlled still images and 6,388 uncontrolled still images. The large still training set contains from 9 to 16 subject sessions per subject, with the mode being 16. The second training set is the 3D training set that contains 3D scans, and controlled and uncontrolled still images from 943 subject sessions. The 3D training set is for training 3D and 3D to 2D algorithms. Still face recognition algorithms can be training from the 3D training set when experiments that compare 3D and still algorithms need to control for training.

The validation set contains images from 466 subjects collected in 4,007 subject sessions. The demographics of the validation partition broken out by sex, age, and race are given in the following figure.

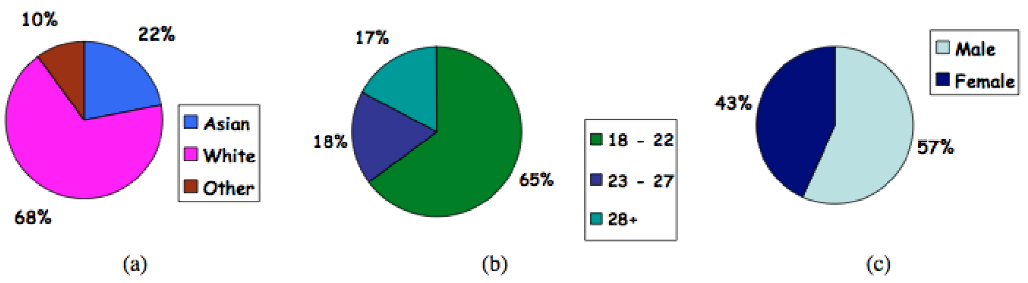


Figure 7-4: Demographics of FRP ver2.0 validation partition by (a) race, (b) age, and (c) sex.

The validation partition contains from 1 to 22 subject sessions per subject (see figure below).

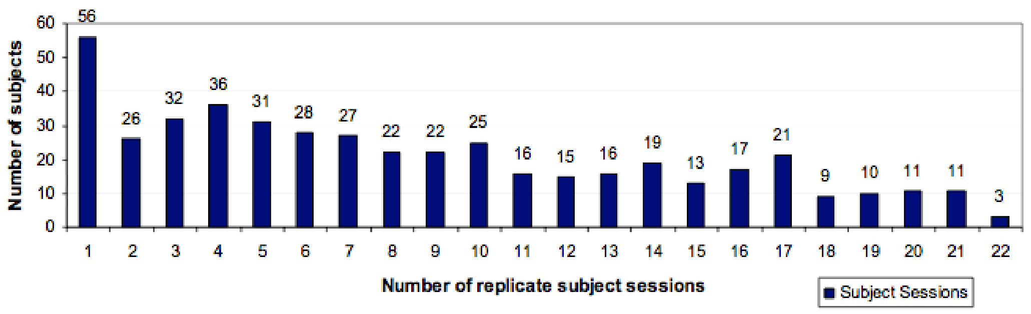


Figure 7-5: Histogram of the distribution of subjects.
X axis represents the number of replicate subject sessions. The histogram is for the ver2.0 validation partition.

7.1.3 Description of the Evaluation Test

The experiments that will be performed to evaluate the EFRS are designed to improve the face recognition algorithm with emphasis on 3D and high resolution still imagery. EFRS will perform six tests:

1. The first test measures the performance on the classic face recognition problem that is the recognition from frontal facial images taken under controlled illumination. To encourage the development of high resolution recognition, all controlled still images are taken in high resolution. In this test, the biometric samples in the target and query sets consist of a single controlled still image. It is clear that multi-still images of a person can substantially improve performance. This test operates in 2D.
2. The second test is designed to examine the effect of multiple still images on performance. In this test, each biometric sample consists of the four controlled images of a person taken in a subject session. The biometric samples in the target and query sets are composed of the four controlled images of each person from a subject session. This test operates in 2D.
3. The third test is focalized on 3D imagery and measures performance when both the enrolled and query images are in 3D. In this test, the target and query sets consist of 3D facial images. One potential scenario for 3D face recognition is that the enrolled images are 3D and the target images are still 2D images.

4 The forth test is designed to measure the progress on recognition from uncontrolled frontal still images. In this test, the target set consists of single controlled still images, while the query set consists of single uncontrolled still images. The “supporters” of 3D face recognition claim that 3D imagery is capable of achieving an order of magnitude of improvement in face recognition performance. Recognizing faces under uncontrolled illumination has numerous applications and is one of the most difficult problems in face recognition.

5 The fifth test explores this scenario when the query images are controlled. The query set consists of a single controlled still. This test operates on 3D Imagery.

6 Finally, test number six examines the uncontrolled query image scenario. The query set consists of a single uncontrolled still and the test operates on 3D Imagery.

7.1.4 Baseline Performance

The baseline performance is introduced to demonstrate that a challenge problem can be executed, can provide a minimum level of performance and a set of controls/feedback for detailed studies and evaluation. The face recognition algorithm based on PCA has been selected as the baseline algorithm.

The initial set of baseline performance results has been provided for test 1, 2, 3, and 4. For test 1, 2, and 4, baseline scores have been computed from the same PCA-based implementation. In test 2, a fusion module is added to handle multiple recordings in the biometric samples. The algorithm is trained on a sub-set of 2,048 images from the large training set. The representation consists of the first 1,228 eigenfeatures (60% of the total eigenfeatures). All images were pre-processed by performing geometric normalization, masking, histogram equalization, and rescaling pixels to have mean zero and unit variance. All PCA spaces have been whitened. The distance in nearest neighbour classifier is the cosine of the angle between two representations in a PCA-space. In test 2, each biometric sample consists of four still images, and comparing two biometric samples involves two sets of four images. Matching all four images in both sets produces 16 similarity scores. For test 2, the final similarity score between the two biometric samples is the average of the 16 similarity scores between the individual still images.

An example set of baseline performance results is given for test 3 (2D versus 3D face recognition) in the following paragraphs. It has been obtained in the previous test performed by independent research team and can be considered as a reference point. The baseline algorithm for the 3D scans consists of PCA performed on the shape and texture channels separately and then fused. Performance scores are given for each channel separately and for the shape and texture channels fused. We also fused the 3D shape channel and one of the controlled still images. The controlled still is taken from the same subject session as the 3D scan. Using the controlled still models a situation where superior still camera is incorporated into the 3D sensor. The baseline algorithm for the texture channel is the same as in test 1.

The PCA algorithm adapted for 3D is based on Chang et al. [7].

The results obtained in the example of baseline verification performance for test 1, 2, 3, and 4 are shown in the above figure. Verification performance is computed from target images collected in the fall semester and query images collected in the spring semester. For these results, the time lapse between images is between two and ten months. The performance is reported on a Receiver Operator Characteristic (ROC) that shows the trade-off between verification and false accepts rates. The false accept rate axis is logarithmic. The results for test 3 are based on fused shape and texture channels. The best baseline performance should be achieved by multi-still images, followed by a single controlled still, and then 3D scans. The most difficult category should be the uncontrolled stills.

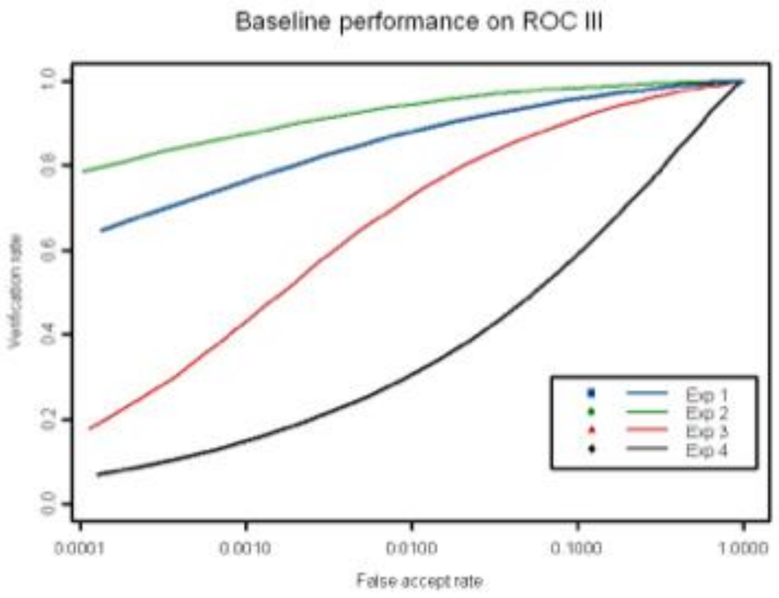


Figure 7-6: Example of expected baseline ROC performance for test 1, 2, 3, and 4

The figure below shows another example of baseline performance for five configurations of the 3D baseline algorithms: fusion of 3D shape and one controlled still; controlled still; fusion of 3D shape and 3D texture; 3D shape; and 3D texture. The best result is achieved by fusing the 3D shape channel and one controlled still image. This result suggests that 3D sensors equipped with higher quality still cameras and illumination better optimized to still cameras may improve performance of 3D systems.

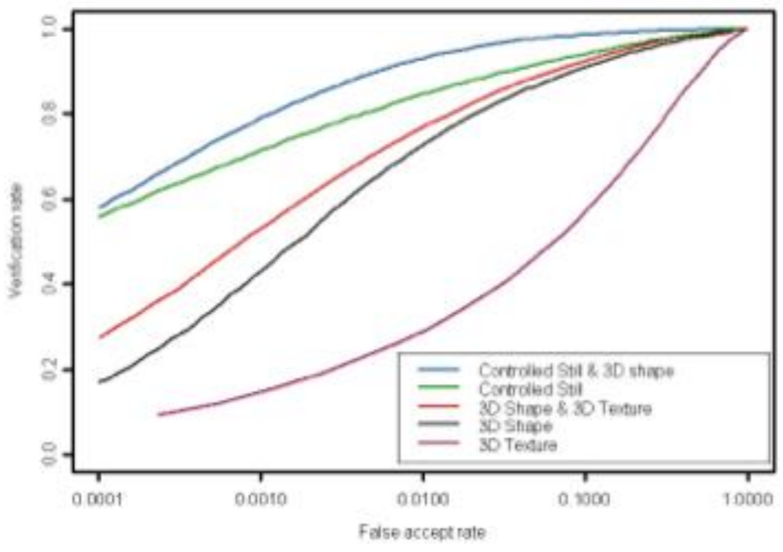


Figure 7-7: Example of baseline ROC performance for Experiment 3 component study

Successful development of pattern recognition algorithms requires that one knows the distributional properties of objects being recognized. A natural starting point is PCA, which assumes the facial distribution has a multi-variate Gaussian distribution in projection space.

In the first facial statistics experiment we examine the effect of the training set size on the eigenspectrum. If the eigenspectrum is stable, then the variance of the facial statistics on the principal components is stable. The eigenspectrum is computed for five training sets of size 512, 1,024, 2,048, 4,096, and 8,192. All the training sets are subsets of the large still training set. The expected eigenspectra should be similar to the ones plotted in the figure below. The horizontal axis is the index for the eigenvalue on a logarithmic scale and the vertical axis is the eigenvalue on a logarithmic scale. The main part of the spectrum consists of the low to mid order eigenvalues. For all five eigenspectra, the main parts overlap.

The eigenvalues are estimates of the variance of the facespace distribution along the principal axes. The figure below shows that the estimates of the variances on the principal components should be stable as the size of training set increases, excluding the tails. The main part of the eigenspectrum is approximately linear, which suggests that to a first order approximation there is a 1/f relationship between eigen-index and the eigenvalues.

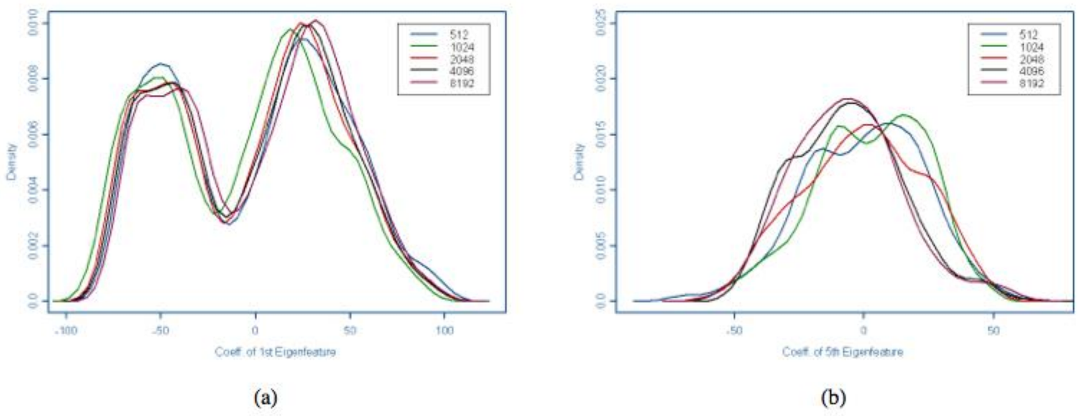


Figure 7-8: Estimated densities

The figure above describes an example of performance on test 1 for training sets of size 512, 1,024, 2,048, 4,096, and 8,192. The figure illustrates the estimated densities for the (a) 1st and (b) 5th eigen-coefficients for each training set (the numbers in the legend are the training set size). To generate the curve label 1024 in (a), a set of images are projected on the 1st eigenfeature generated the 1024 training set. The set of images projected onto the eigenfeatures is a subset of 512 images in common to all five training sets. All other curves were generated in a similar manner. Verification performances at a false accept rate of 0.1% is reported (vertical axis). The horizontal axis is the number of eigenfeatures in the representation. The eigenfeatures selected are the first n components. The training set of size 512 approximates the size of the training set in the FERET Sep96 protocol. This curve approximates what was observed by Moon and Phillips [5], where performance increases, peaks, and then decreases slightly. Performance peaks for training sets of size 2,048 and 4,096 and then starts to decrease for the training set of size 8,192. For training sets of size 2,048 and 4,096, there is a large region where performance is stable. The training sets of size 2,048, 4,096, and 8,192 have tails where performance degrades to near zero.

The examples described in this section allowed to identify and demonstrate the two most important consequences that we expected from the experiments: first, it is evident that increasing the training set increases also the performance of the recognition, and second, it is clear that the selection of the cut off index is not critical.

7.2 Smart card and access delegation

The proposed module will be verified through an experimental test bed. The basic parameters that will be validated and verified through the demonstrator are the followings:

- The introduced overhead (time and processing) of accessing “labeled” rooms is insignificant in the context of the provided security services. To measure the introduced overhead we developed a simulation model that generates multiple requests that follow the poison distribution. This way, we approach system's real time behavior and identify with accuracy the introduced overhead.

- The accuracy of identified people using biometric templates stored in the smart card is high. This corresponds to determine whether the biometric template in the smart card matches with the live captured biometric data. To achieve this goal will we rely on the tests described in Section 7.1.3.

Furthermore, the smart card solution will incorporate the appropriate interfaces to other system modules (e.g., such as accessing the biometric templates through the provided cameras) in order to demonstrate its functionality. The following picture depicts the high level architecture for the required interfaces.

7.3 Dependable Distributed Computation Framework

The role of the framework within the scenario is related to the deployment and execution of the people identification algorithms.

As such, there are three verification phases for the framework in respect to the demonstrator:

1. Setup: a successful installation of the framework on all the related nodes;
2. Deployment: the scenario application can be successfully prepared for execution;
3. Execution: the framework is able to process the scenario application even in the presence of faults.

In the following, we will describe in more detail the verification requirements associated with the above phases.

7.3.1 Setup

In the setup phase, we need to validate the platform, by guaranteeing that all the nodes required by the demonstrator platform are capable of handling the framework. Most of such validation has already been done on the chosen embedded node, i.e., the BeagleBoard XM board. The validation will be extended to the ETH SecuBoard when the final prototype will be available. In particular, the requirements involved the following:

1. To be able to run the Java Virtual Machine, due to the framework being written in Java;
2. To be able to run a wide range of compilers, namely the Java Compiler, GNU make and Clang, in order to support the majority of the compilation flows for applications;
3. To have the libraries required by the scenario application, if any.

Finally, the actual framework installation will be tested as soon as completed. However, we envision no particular issues as long as the above requirements are met.

Please note that these requirements can be considered test-relevant only on embedded-class nodes. On this regard, since the framework benefits from the presence of high-availability server(s), the testing platform will comprise at least one server node and one client node; a client node is any PC with remote connectivity. These nodes, however, are not considered critical due to their high computational capabilities and wide support for the required software dependencies.

On all nodes, we expect to be running a GNU/Linux distribution. However, the framework is designed to run on any operating system supporting the Java Runtime.

7.3.2 Deployment

Deployment in our framework is equivalent to autonomous download of artifacts from versioned repositories. The verification involved in such a phase comprises the following:

1. The repositories can be accessible from the nodes, in particular in terms of authorization if secure access is requested;
2. The artifacts related to code can be compiled/linked successfully, if necessary;
3. The artifacts related to code can be loaded successfully, in preparation for execution.

Even if the scenario application is the reference one, these verification steps can be conducted on a sample application with no loss of generality. The only issues that can stem regard custom libraries used by the specific application, which is actually a setup concern.

7.3.3 Execution

In the execution phase, the nodes have already started the runtime processes and can accept payload from the scenario application. Here we must verify the following:

1. The framework autonomously handles computation, with no communication issues;
2. In the presence of faults, the execution either proceeds with limited resources, or pauses while waiting for new resources.

The verification of the first point is rather straightforward and it will be conducted on sample applications before the final scenario application has become available.

For the second point, we will monitor the evolution of the computation after the introduction of random user-generated faults. Such faults will include the temporary disconnection of certain nodes, but also the simulation of critical communication issues that may prevent the recovery of execution. Since the framework is designed to tolerate any kind of fault as long as the consensus at the persistence nodes is preserved, our objective is to guarantee that execution is never aborted. It is instead paused as long as completion is temporarily infeasible.

8 Conclusions

In this deliverable, we presented the integration and validation plan for the “People Identification at the Stadium” use case and demonstrator.

Starting from the description of the people identification application, we provided an overview of the involved technologies and we illustrated the solution proposed for three real-world scenarios.

Finally, the document presents the integration, verification and validation plan, in order to demonstrate and validate the nSHIELD framework in the reference application.

9 References

[1] M. Turk and A. Pentland, “Eigenfaces for Recognition,” J. Cognitive Neuroscience, vol. 3, no. 1, pp. 71 - 86, 1991.

[2] L. Sirovich and M. Kirby, “Low-Dimensional procedure for the characterisation of human faces,” Optical Society of America, vol. 4, no. 3, pp. 519- 524, 1987.

[3] M. Kirby y L. Sirovich, «Application of the Karhunen- Loève procedure for the characterisation of human faces,» IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 12, pp. 831-835, 1990.

[4] D. Liu, K. Lam and L. Shen, “Optimal sampling of Gabor features for face recognition,” Pattern Recognition Letters, vol. 25, no. 2, pp. 267 - 276, 2004.

[5] H. Moon and P. J. Phillips, “Computational and performance aspects of PCA-based face-recognition algorithms,” *Perception,* vol. 30, no. 3, p. 303–321, 2001.

[6] D. Liu, K. Lam and L. Shen, “Illumination invariant face recognition,” *Pattern Recognition,* vol. 38, no. 10, p. 1705–1716, 2005.

[7] H. Kyong, I. Chang, K. W. Bowyer and P. J. Flynn, “An evaluation of multi-modal 2d+3d face biometrics,” *IEEE Trans. PAMI,* vol. 27, no. 4, pp. 619 - 624, 2005.

[8] Face Recognition Vendor Test (FRVT) 2002, http://www.nist.gov/itl/iad/ig/frvt-2002.cfm

[9] B. Sivakumar Tati, A framework to implement delegation in offline PACS, KTH, Sweden: M.Sc. degree thesis, 2012.

[10] ISO/IEC 7816 part 1-15, Available on-line: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=29257

[11] C. Lambrinoudakis, “Smart card technology for deploying a secure information management framework,” *Information Management & Computer Security,* vol. 8, no. 4, p. 173 – 183, 2000