FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Confidential Desktop

Towards an access control framework for preserving data confidentiality based on environmental conditions

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Medieninformatik

eingereicht von

Sophie Weiß BSc
Matrikelnummer 0925295

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Privatdoz. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Edgar Weippl

Wien, 17. Dezember 2015

Sophie Weiß

Edgar Weippl

Confidential Desktop

Towards an access control framework for preserving data confidentiality based on environmental conditions

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Media Informatics

by

Sophie Weiß BSc
Registration Number 0925295

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Privatdoz. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Edgar Weippl

Vienna, 17th December, 2015

Sophie Weiß

Edgar Weippl

Erklärung zur Verfassung der Arbeit

Sophie Weiß BSc
Hetzgasse 15/9
1030 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 17. Dezember 2015

Sophie Weiß

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Edgar Weippl for the continuous support of my master thesis. A big thank goes to Fuensanta Torres Garcia for all the useful comments and advice when I was lost. I also want to thank my colleagues and friends, Christoph H., Peter K., Katharina K. and Simon S., who have supported me during my thesis.

I would like to thank my parents for their encouragement and love throughout my entire life. Without their financial and mental support during my studies, this thesis would not have been possible.

A special heartfelt thank I wish to express my dearest friend and fellow student Dani S.. Every time you have an open ear for me and your smile saves me even through the darkest of times. Our friendship is for life.

Finally, my beloved sweetheart, Peter. Words cannot express my deep gratitude I feel for you. Your unconditional love makes my life worth living. I love you with all my heart. Thank you for everything.

Kurzfassung

Vor allem in sicherheitskritischen Branchen wollen Unternehmen sicherstellen, dass unternehmensinterne Daten vor unautorisierten Augen verborgen bleiben, zum Beispiel wenn externe Mitarbeiter wie Reinigungspersonal kritische Bereiche betreten. Wenn ein erhöhter Sicherheitsstandard im Büro verpflichtend ist, muss sichergestellt sein, dass nicht autorisierte Personen keine Möglichkeit haben, Zugang zu vertraulichen Daten, die auf dem Computerbildschirm angezeigt werden, zu bekommen.

Diese Arbeit beschreibt einen Ansatz für die Erhaltung der Privatsphäre durch die Implementierung eines kontextbezogenen Zugriffskontrollsystems namens *Confidential Desktop*, das kontinuierlich überprüft, ob der Benutzer allein im Büro ist oder nicht. Dies passiert mittels Erkennen von Aktivitäten von Eindringlingen durch die Verwendung von verschiedenen Funktionen für die Erkennung von Gesichtern, Bewegung, Klängen und Bluetooth-Geräten. Es wird in simulierten Szenarien gezeigt, dass auch mit bestehender Standard-Hardware und State-of-the-Art Methoden Aktivitäten von Eindringlingen durch das System in Echtzeit entdeckt werden können. So ist es mit der präsentierten Lösung möglich, den Zugriff auf angezeigte Daten zu steuern und die Vertraulichkeit sensibler Daten vor neugierigen Blicken zu schützen.

Abstract

Especially in security-critical industries, companies want to ensure that company-internal data remain hidden from unauthorized eyes, e.g., if external workers like cleaning staff are entering critical areas. If an increased security standard is mandatory at the office, it must be ensured that non-authorized persons have no possibility to get access to confidential data, which is displayed on the computer screen.

This thesis describes an approach for preserving screen privacy by implementing a context-aware access control framework called *Confidential Desktop*, which continuously checks whether the user is alone at the office or not. This is done by detecting predefined intruder's activities by the use of different features like detecting faces, motion, sounds and bluetooth devices. The feasibility is shown in simulated scenarios that even with existing standard office hardware and state-of-the-art methods, intruder's activities can be detected by the framework in real-time. Thus the proposed solution can control the access to displayed data and preserve the confidentiality of sensitive information from prying eyes.

Contents

CHAPTER 1

Introduction

1.1 Problem Definition

Confidentiality is a key concept of information security and means in the context of this thesis the ability to keep sensitive data, which is displayed on the computer screen, private. Particularly in security-critical industries, companies want to ensure that company-internal data remain hidden from unauthorized eyes, e.g., if external workers like cleaning staff or repairmen are entering critical areas. Another example would be the case of a bank employee, who receives customers in his office for advice. While a customer is in the room, access to security-critical information should be denied. Similarly, the opposite might be the case that certain security criteria require that two people have to be in the room in order to have access to system resources ("four-eyes principle").

Especially if an increased security standard has to be maintained, it must be ensured that non-authorized persons do not have the possibility to get access to confidential data. The case of an intruder coming into a critical environment and asking the employee for sensitive information (social engineering [Tho04, Wor07]) would be impossible if the software would only grant access to this information, if the employee was alone.

In certain scenarios, employees or even private users want to protect displayed information on their computer screen from prying eyes - this is mostly tried by manually hiding opened windows or locking screens, when somebody is coming into the visual field of the computer screen. But sometimes the users are not able to react in time, which often leads to information leakage and awkward situations.

Since there is a substantial offer of privacy screens and physical filters (e.g., screen protectors) available on the market, it seems that users today in general want increased protection of their computer screens from prying eyes. Privacy screens and filters help to keep the displayed information private by showing nothing but a blank, black screen when it is viewed from an angle other than the desired view.

Such solutions could be expensive, or when they, e.g., need to remove the polarising filter of a display, irreversible. Therefore they are particularly unsuitable for most business scenarios.

This thesis describes an approach for preserving screen privacy by implementing a context-aware access control framework called *Confidential Desktop*, which continuously checks whether the user is alone in the office or not.

This is achieved by detecting intruder's activities by utilizing recent state-of-the-art techniques in the area of computer vision and media informatics for detecting humans, motions or sounds.

1.2 Methodology

The aim of this thesis is not only to present theoretical approaches, but also to provide a feasible implementation in order to achieve a reasonable technique. The methodological approach consists of the following parts:

- **Literature review** Background information about state-of-the-art approaches in areas like computer vision (Section 3.1) and audio analysis (Section 3.2), which are suitable as background methods for the framework, were gathered and already existing solutions for access control (Chapter 2) were analyzed.
- **Prototype implementation** The framework *Confidential Desktop* was implemented as desktop application using Java, including a graphical user interface. Based on the gathered information, different exemplary background features like face and motion detection were implemented and integrated dynamically into the framework to continuously detect, if an intruder is coming into the room.
- **Scenario evaluation** For the evaluation of the framework experimental tests in three different simulated scenarios where an intruder enters the room were performed to measure the detection quality of each integrated feature and further to identify the best combination of the utilized algorithms.

1.3 Structure of the Thesis

The first chapters provide an overview of state-of-the-art, i.e., on different access control approaches (Chapter 2), on approaches in the area of computer vision and image processing (Section 3.1) and audio analysis (Section 3.2). The last two are intended to show the amount of possible approaches for detecting the presence of one or more persons in a room, which could be used as features for the framework.

Chapter 4 describes the system design and implementation of the proposed framework *Confidential Desktop* and further of the integrated exemplary background features (Section 4.4): Bluetooth Device Discovery, Face Detection, Motion Detection, Noise Detection and Smartphone Extension: Motion Detection.

The exact structure and workflow of the performed scenarios as well as the evaluation results of the framework are discussed and presented in Chapter 5. In the end, the contribution is summarized, found results are outlined and an outlook on possible future work is provided (Chapter 6).

CHAPTER 2

Access Control

This Chapter describes and discusses the fundamentals and different policies of access control. The first Section explains the principles and the general access control mechanism. The second part of this Chapter moves on to describe in greater detail the different access control policies, respective literature and how the mentioned approaches are related to the proposed framework *Confidential Desktop*. The last Section summarizes and compares the findings and illustrates them in a table.

2.1 Fundamentals

Access control is “the prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner” [fSG89]. It is one of the main goals in computer security and “critical to preserving the confidentiality and integrity of information” [FKC03]. Access Control has three main objectives:

- Prevent unauthorized users to access an asset, e.g., a guest that uses a public company Wireless Local Area Network (WLAN) cannot access protected files in a company network

- Prevent legitimate users to access assets in an unauthorized way, e.g., a non-admin user has permission to read configuration files but is prohibited to change them

- Enable legitimate users to access resources, e.g., a user with role *Dean* has all permissions assigned to this role within an university network

Furthermore, three different, basic access control elements are defined, namely *subject* (an entity that can access objects, e.g., user, process, etc.), *object* (an access controlled resource, e.g., files, directories, etc.) and *access right* (the way in which a subject accesses an object, e.g., read, write, execute etc.) [SB08].

As shown in Figure 2.1 the general access control mechanism combines essential functions of the field of computer and information security like identification, authentication, and authorization. First a user has to identify and authenticate him- or herself, i.e., he or she has to make and verify a claim of identity. This can be done for example by entering the correct password (something the user knows), by using an ID card (something the user has) or by biometric authentication like face recognition (something the user is). After the user has been successfully identified and authenticated, the authorization function controls access to system resources, managing to which resources the user has what kind of access (read, write, etc.).

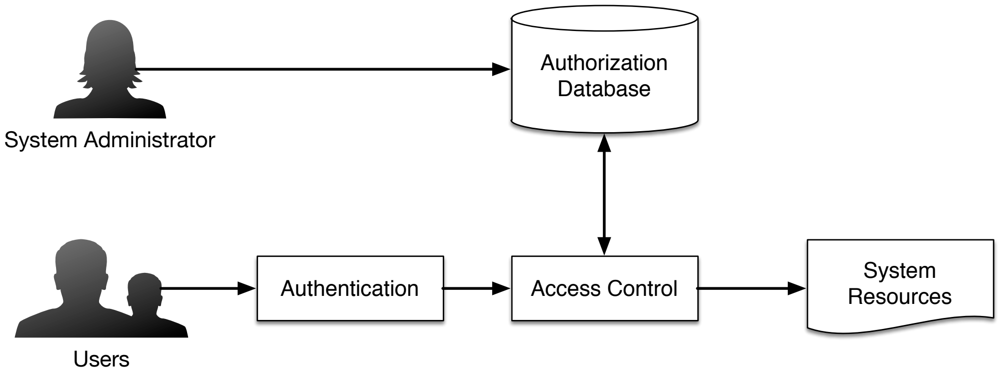


Figure 2.1: General Access Control Mechanism

While the three steps illustrated in Figure 2.1 constitute the foundation of access control, access to system resources might be restricted in various ways. So called access control policies describe access control mechanics, e.g., based on roles, context or attributes of users and resources. The following Section will address different access control models in detail, which will be further summarized and compared with each other at the end of this Chapter. In addition, the discussed approaches are analyzed with respect to their applicability for the proposed framework.

2.2 Access Control Models

Since the beginning of research in the area of access control, in 1960 and 1970 [FPS11], a variety of different policies were developed. Before the 1990s, where Role-based Access Control (RBAC) was proposed [FKCB92], Discretionary Access Control (DAC) and Mandatory Access Control (MAC) were "the most popular access control concepts" [FPS11]. This Section describes and discusses these and other access control policies. In the end of every Subsection, their applicability for the proposed framework

Confidential Desktop is analyzed and as part of that, some of the advantages of the policies are extracted.

2.2.1 Discretionary Access Control

One of the most common policies of access control is Discretionary Access Control (DAC) [QZW+85]. DAC determines, that the access to system resources is controlled by its ownership, i.e., the access to resources is controlled based on the identity of the user, who wants to access the resource [SB08].

A common approach for DAC is the access matrix by Lampson et al. [Lam69, Lam74]. A simple example of such an access matrix is illustrated in Table 2.1.

		OBJECTS			
		File 1	**File 2**	**File 3**	**File 4**
SUBJECTS	**User A**	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

Table 2.1: Access Matrix [SS94, SB08]

One dimension of the access matrix defines the subjects, respectively the entities, which want to access the resources. Instead of individual users, these entities could be user groups or processes as well. In the other dimension of the matrix, the objects are defined, i.e. the resources whose access is controlled. In this specific access matrix, it is determined, that, e.g., User A owns File 1 and File 3 and has both read and write access to them.

However, while the access matrix as a conceptual abstraction for the illustration of access control rules is well suited, Sandhu et al. [SS94] consider that especially for large systems the implementation of such a matrix would be very memory consuming and usually many cells would be left empty. For this reason, they propose some methods to implement the access matrix in practice.

One approach is the implementation by the use of Access Control Lists (ACLs). For this method the access matrix is decomposed by its columns. This approach is well suited if the access rights for an object should be determined, i.e., which user has which access for this resource. For the example mentioned above (Table 2.1), the ACLs for File 1 and File 3 would be as visualized in Figure 2.2. For each file the users and their according access rights are listed. For example for File 1 it is stated, that User A (the first element in the list) owns this file and has read and write access. User B, the second element in the list, only has read access. User C has read and write access.

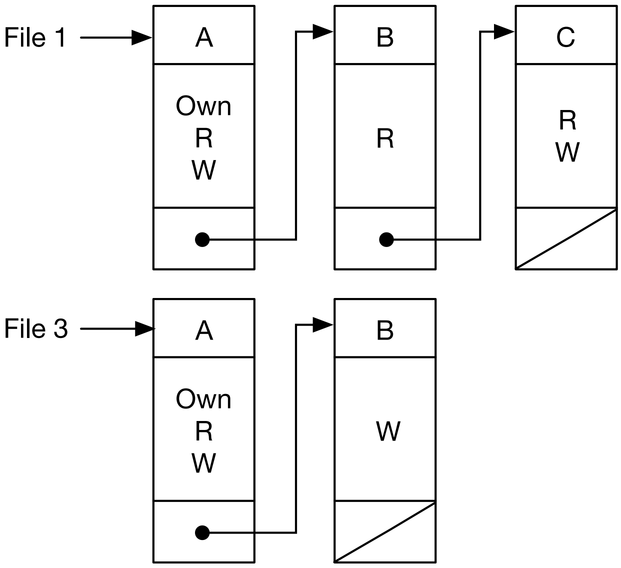


Figure 2.2: ACLs [SS94, SB08]

Another approach, which is similar to ACLs and also proposed by Sandhu et al. [SS94], is based on capability lists. These lists result from a decomposition of the access matrix by its rows and contain the access rights for every object of a specific user. Unlike ACLs, this approach is well suited for determining the access rights of a user for all resources. To stay with the example, Figure 2.3 shows the capability list of User A, which contains the already mentioned access rights for File 1 and File 3.

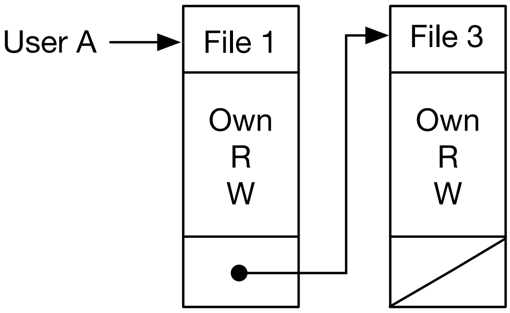


Figure 2.3: Capability lists [SS94, SB08]

Table 2.2 shows an example of an authorization table, a data structure also proposed by Sandhu et al. [SS94]. This authorization table contains all access rights for every subject for all objects in one row each. So for User A there are six rows, 3 different access modes for each File 1 and File 3. As already mentioned by the authors, an authorization table is very well suited to be implemented in a relational database.

Subject	**Access Mode**	**Object**
A	Own	File 1
A	Read	File 1
A	Write	File 1
A	Own	File 3
A	Read	File 3
A	Write	File 3

Table 2.2: Authorization Table [SS94, SB08]

Based on the identity of the person who enters the room, DAC could be used to provide an additional access control method for the framework *Confidential Desktop*. Integrating an advanced feature for authentication into the framework would make it possible to identify a person who comes into the room. After that it could be checked in the background, which access rights for this person are defined for specific programs or resources in general. For example, if the user has opened a program or file on his computer screen, for which he or she is allowed to see and modify it and there is someone entering the room who has the same rights for this resource, then it does not have to be closed or hidden from the framework. So the framework has implemented the access rights in the background, for example as authorization table in a database and is able to check the rights of a person who comes into the room at any time. Because of the high flexibility of DAC, it is very well suited for the use in the framework. Owners of files could determine who is allowed to see this files and whether they should be hidden from the framework if other persons enter the room.

In general, especially in the private use of the framework, the user as admin should have the possibility to specify, which resources should be hidden from the framework if someone or a particular person enters the room. So, for example, if the chef enters the room, a private text file should be hidden, because he should not have any access to this file. The colleagues from next room have read access to the file, so it does not have to be closed, if they enter the room.

2.2.2 Mandatory Access Control

Another policy of access control is Mandatory Access Control (MAC) [Lin06]. Before RBAC (see following Subsection 2.2.3), DAC and MAC were “the most popular access control concepts” [FPS11]. In contrast to DAC, where the access to a resource is controlled by its ownership, at MAC the access is controlled by the operating system. Means that, based on Multilevel Security (MLS) it is determined by different security levels, e.g. “confidential”, which could be a lower level of security than, e.g., “top secret”, how critical the system resources are. Subjects then have a specific security clearance of a given level. Which and how many security levels are used, which level the resources are classified with and which level of clearance the subjects have, is stated by the administrator in the authorization database (see Figure 2.1). Further, in order to preserve the confidentiality, two security properties are needed:

- **No read up** A subject, respectively a user, is only able to read an object of less or equal security level. For example, a user, who has the clearance “confidential” is not able to read a resource, which is classified as “top secret” but a resource, which is classified with the level “confidential” or lower.

- **No write down** A subject is only able to write, modify, or delete an object of greater or equal security level. For example, a user, who has the clearance “confidential” is allowed to modify a resource with the security level “top secret”.

The second property is needed because of the following reason. The term *mandatory* in MAC means, that the user is not allowed to override the security policy or to enable the access to a resource for other users. Assuming that a user, who has the clearance “top secret” is allowed to read a resource, which is classified as “top secret” *and* has the permission, to modify a resource with the security level “confidential”, the user would have the possibility to release secret information from a higher security level to a lower level and thus to other users, who only have the lower clearance “confidential”.

MAC is stricter than DAC in terms of allocation of access rights. Generally the use case has to be considered that the user does not have any admin rights for the framework, for example, if the framework is used in a company and the user has to use it as an employee. In this case the user is not able to specify any access rights, because they would have been defined by an admin in the company. From the viewpoint of a company with many employees it can be a useful option to use an additional MAC model with the framework because for every subject and every resource just one security level has to be specified. In any case, just as with DAC, an advanced feature for authentication has to be integrated to identify the person, who is entering the room or coming closer to the workstation. If such a feature is not available, a simple solution in combination with *Confidential Desktop* would be to define a particular security level for the framework, at which it is intended to regulate access. I.e., starting with the security level "secret", the access to all resources which were classified with this or a higher level should be denied by the framework when an intruder enters the room, resources with lower levels remain unaffected. Another option would be to go one level deeper behind the framework itself and use the MAC model to specify what admins, other (particular) users and guests are allowed to do with the framework. E.g., admins have the possibility to customize access rights for special subjects or objects, ordinary users are allowed to determine access rights for their own resources and guests only are able to specify, if applications should be closed or just hidden by the framework.

In general, it must be said that for the proposed framework *Confidential Desktop*, which should itself be a highly flexible solution, MAC is not flexible enough for other access control models like DAC or Attribute-based Access Control (ABAC) (see Subsection 2.2.4) should be preferred.

2.2.3 Role-Based Access Control

Since the early 90s, researchers are intensively examining the concept of an access control model based on roles for administrating file permissions and restricting access to authorized users. Among others, Ravi Sandhu [SCFY96, San98] and David Ferraiolo [FK09, FKC03] are prominent in the literature on Role-based Access Control (RBAC) models.

The main purpose of RBAC is to define access permissions based on roles and then assign one or more roles to a subject. This is also one of the main advantages of RBAC, because access permissions are defined for a small number of roles rather than for every single user, which reduces cost and complexity.

In 2000, the National Institute of Standards and Technology (NIST) developed a standardized *NIST RBAC model*, consisting of a combination of both, the approaches of Sandhu et al. and Ferraiolo et al. [SFK00, FSG$^+$01].

The *NIST RBAC model* describes four different levels of RBAC: Flat RBAC, Hierarchical RBAC, Constrained RBAC and Symmetric RBAC, whereby the first level, Flat RBAC (Figure 2.4), describes following basic rules [SFK00]:

- users acquire permissions through roles
- must support many-to-many user-role assignment
- must support many-to-many permission-role assignment
- must support user-role assignment review
- users can use permissions of multiple roles simultaneously

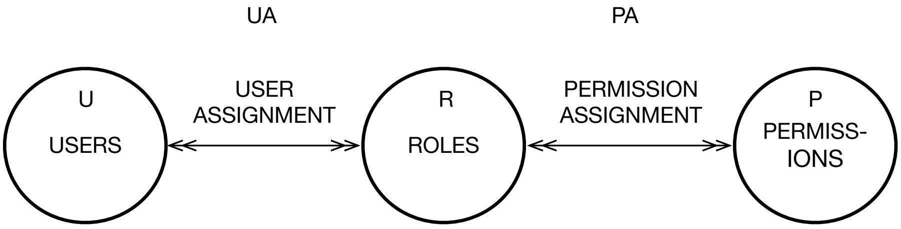


Figure 2.4: Flat RBAC [SFK00]

The second level, Hierarchical RBAC (Figure 2.5) adds an additional rule, namely a hierarchy of roles must be supported. For example, the role structure for a company's network could be defined as follows: above the role *Guest* there may be the role *Employee*, which inherits all permissions from the role *Guest* and above *Employee* there may be the role *Admin*, which inherits all permission from the role *Employee.*

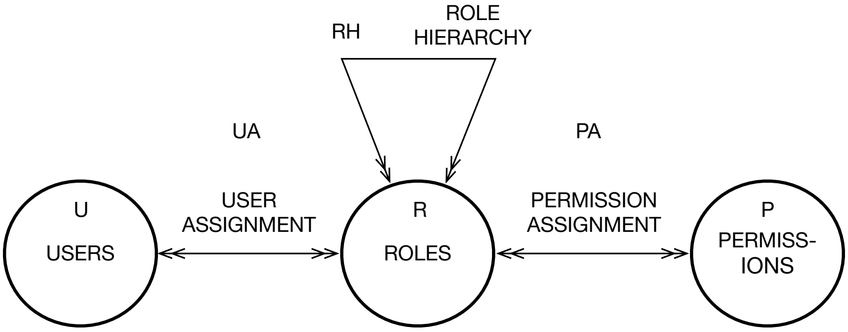


Figure 2.5: Hierarchical RBAC [SFK00]

Next to the NIST standard, a huge amount of other approaches has been published over the last years, all going into different directions. Fuchs et al. [FPS11] analyzed in a comprehensive survey, consisting of other surveys among other things, the development of RBAC models in recent years, classify their findings and further discuss trends in research. For their survey, the authors developed an automatic search engine to search for relevant publications in various databases, e.g., the ACM Digital Library[1] and Google Scholar[2]. The final result of this automated search was the assembly of 1361 publications from 1992 to 2011. Through a multi-stage classification process, which includes, among others, information about author, title and structure of the papers, the found publications were classified in 32 different research areas. These research areas were grouped by the authors in 3 major areas, namely: (i) early RBAC publications and publications with (ii) theoretical and (iii) applied focus. The smallest publication classification group was the first one, which only consists of 15 early RBAC publications. The remaining publications were distributed balanced on both theoretical (704 publications) and practical focus (642 publications). Furthermore, the authors analyzed their findings in terms of their year of publication and found an increase in publications over the years. While in 2000 43 publications in the research area of RBAC models were published, there were in 2010 already 159 publications. From the author's perspective, RBAC will stay a hot topic in the future and the research in this area is about to specialize even further in different topics and also will react on changes in various sectors such as e.g., social media and cloud computing.

---

[1] http://portal.acm.org/dl.cfm [Online; accessed 12.12.2015]

[2] https://scholar.google.at [Online; accessed 12.12.2015]

Just as DAC and MAC, RBAC provides a great way to incorporate an additional mechanism for access control into the proposed framework *Confidential Desktop*. On the one hand, various roles make it easier to grant privileges for a large amount of different stakeholders. With the help of specially integrated features for recognition and authentication of individuals, the behavior of the framework may vary at different roles. So depending on whether the person entering the room is recognized by the framework as a superior or cleaning staff, it reacts differently. The detection or identification of individuals can happen with the help of special face recognition or other biometric identification such as e.g., gait style, in combination with machine learning algorithms. Also a Radio-frequency Identification (RFID) tag in an identification badge or card can be used to authenticate persons, who enter the room. If the behavior of the framework, in the case it detects an intruder, should be changed, it is less effort to change the behavior for e.g., one or two roles, instead of change it for each individual user. This saves time and makes the management of access rights easier.

On the other hand, as already discussed in MAC, RBAC also can be used to control access to the framework itself. A user, who owns the role Admin is allowed to customize the framework for his private use or for a company. A user, who has the role Employee and uses this framework in the office, is only permitted to execute the framework, but not to modify it.

Before proceeding to examine another access control model, it will be necessary to mention that access control policies like the already discussed models DAC, MAC and RBAC do not have to be exclusive. As already noted by Sandhu et al. [SS94], they “can be combined to provide a more suitable protection system”.

2.2.4 Attribute-Based Access Control

Attribute-based Access Control (ABAC) [HFK+14] is a relatively new area of research and considers, contrary to the access control models mentioned so far, also information about the resources or relations between subjects and resources. While DAC, MAC and RBAC are user-centric, the ABAC model also takes additional, predefined attributes like resource types or environmental conditions into consideration. There are three different types of attributes at ABAC [SB08]:

- **Subject attributes** Attributes, which characterize a user, process or application (see explanation of the term Subject in Section 2.1) like identifier, name, title, date of birth, company, job position, role, etc.

- **Object attributes** Attributes, which characterize a system resource, e.g., a file or directory (see explanation of the term Object in Section 2.1) like title, type, location, ownership, etc.

- **Environment attributes** Attributes, which describe environmental conditions and context-based information, like e.g., current date, day time, weather, location of the user etc.

A big advantage of using ABAC as an access control model is the high flexibility of the model. Any number of various attributes can be combined to define a rule set and to adapt the model for different situations. Depending on which attributes are used, it can be said that RBAC and context-aware access control are special forms of ABAC.

Especially the case of including contextual information about environmental conditions for context-aware access control is relevant for the proposed framework and discussed in detail in the next Subsection 2.2.5.

An additional inclusion of characteristics of users and access controlled resources also makes a lot of sense for the proposed framework *Confidential Desktop*. Through the possibility to know at any time what person is sitting before the computer as user or further what other people enter the room, recognized properties in addition can be included into the access control mechanism. The framework could consider, e.g., the job position of the person entering the room and thus behave differently, depending on whether it is the boss or colleague, who is coming closer to the workstation (comparable to the use of roles in RBAC, Subsection 2.2.3). Also the type of resource being accessed, may decide how the framework in which case should behave. For example can be determined that programs should be closed, browser only minimized, videos muted, etc. Another possibility would be to take the location of the user into consideration when using the framework. If the user is e.g., sitting in the office at work, all computer game applications should close immediately, if somebody enters the room. If the user is sitting at home, the applications should only be paused by the framework.

2.2.5 Context-Aware Access Control

There is a rapidly growing amount of literature discussing various approaches for context-aware access control, which include e.g., environmental conditions into the process of dynamically granting or denying access to system resources.

A crucial keyword here is ubiquitous or pervasive computing [Nie07], a concept which, for example, is used in home automation or so called "smart homes" which comes up with powerful tools for gathering data - anywhere and at any time.

Covington et al. [CLS$^+$01] discuss in their work the above mentioned topic by utilizing sensors at home for establishing a context-aware access control system. Based on the description of hierarchical environmental roles (Figure 2.6) - similar to the idea of subject roles - they extend the traditional role-based access control model. Apart from days of the week, environment roles can also be defined by the current temperature, day time or identity of a subject ("adult", "child", or "pet"). Furthermore the authors describe various possible use cases, which show the potential of using context-aware access control systems, e.g., children are allowed to watch television only between 9 p.m. and 10 p.m., or the babysitter has access to the fridge on work days only.

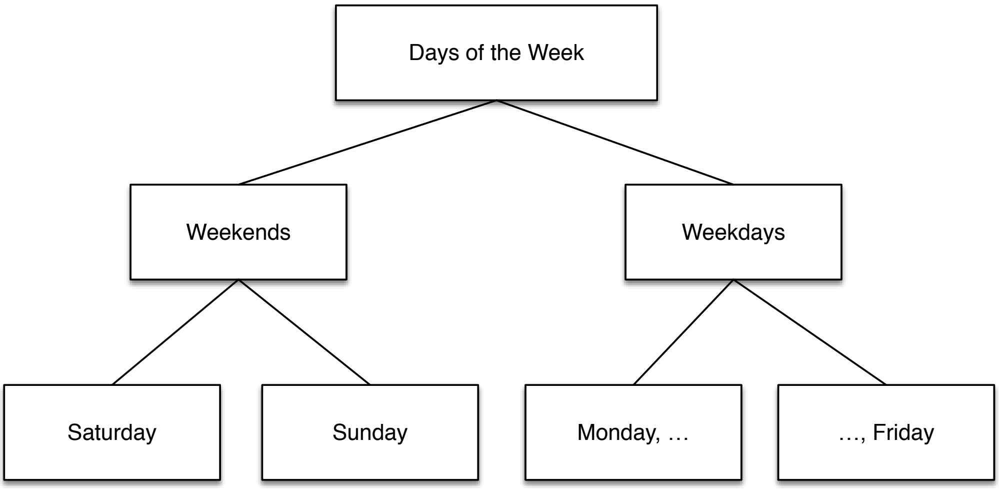


Figure 2.6: A Simple Environment Role Hierarchy [CLS+01]

Drawbacks of this approach are presented in the work of Zhang et al. [ZP04], who hold the opinion that the definition of environment roles “may not be feasible in practice because the potential large amount of environment roles make the system hard to maintain”, further they argue, that this would lead to a loss of the advantages of RBAC.

Instead of environment roles, the authors propose to switch the permission roles of a user dynamically based on, e.g., current location information, which is realized using “Role Hierarchy State Machines” (Figure 2.7). For example, if Linda logs in to her user account in her office, she gets assigned to the role *Professor*, which means that she has read and write access to her files. When she leaves her office, a transition in the role state machine is triggered and her role is changed to *Faculty* (Read Only), so Linda will not be able to write files any more when she is outside her office.

Toninelli et al. [TMKL06] address the challenge of two individuals sharing resources with each other by using mobile devices. They portray the issue that traditional access control models focus on the entity’s identities and roles rather than on the respective context.

Based on the example, mentioned in their paper, a “meeting occurring during a conference among members of different universities working on a common project” the authors expose detailed access control challenges and describe a semantic context-aware policy model for this scenario.

A valuable approach related to this thesis is discussed by Kulkarni et al. [KT08] and outlines following scenario:

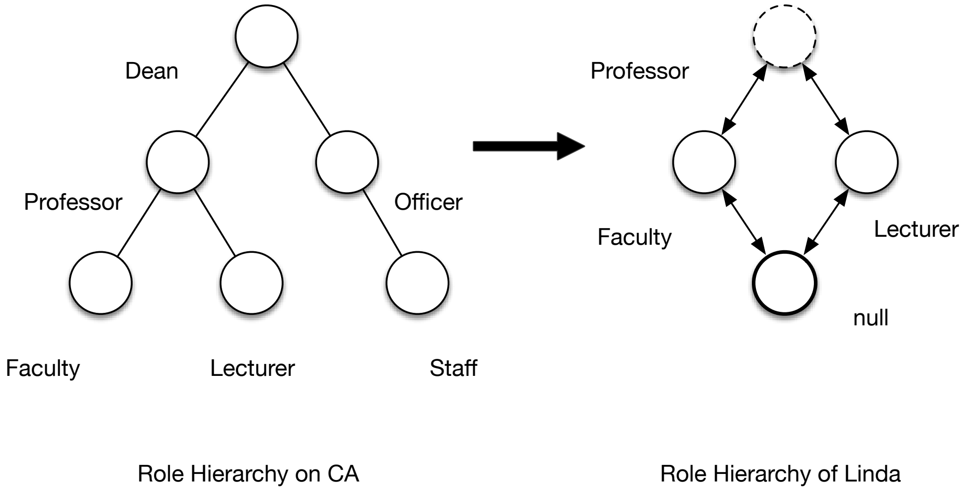


Figure 2.7: Role Hierarchy of Central Authority (CA) and Linda [ZP04]

An application running on a user's device plays music. If the user goes into a certain room, a location-based event is triggered. If the user is alone in this room, the application, e.g., gets access to automatically play music on a music player. If a second person enters (or the user leaves the room) the application's access to the music player is revoked and played music is resumed on the user's mobile device.

For this purpose the authors defined a context-aware RBAC (CA-RBAC) model and developed a role-based framework which allows to program secure context-aware pervasive computing applications like the previously mentioned music player application.

These were just a few examples of different approaches for establishing a context-aware access control model. Also for the framework *Confidential Desktop*, which is proposed in this thesis, the goal is to integrate an ubiquitous, context-aware access control mechanism into the every day working environment of an user. Based on the detection of intruder's activities in the room, the access to system resources and applications is limited by the framework.

Of course, next to the number of detected persons in the room, there are many other possibilities to include environmental conditions and contextual information into such a framework. Dependent on information like e.g., day of the week, current time, sensors or additional devices near the workstation, sound level in the office, etc. the framework and thus the access control mechanism could change its behavior. This is one of the big advantages of attribute-based, respectively context-aware access control models, namely that they are high flexible and adaptable for an unlimited number of different scenarios. The main focus of the proposed framework is to develop an equally flexible solution to allow further opportunities for the access control mechanism to include additional information about environmental conditions.

2.3 Summary and Comparison

So far this Chapter has analyzed various access control models in detail and discussed their applicability for the framework *Confidential Desktop*, which is proposed in this thesis, in a comprehensive way. This Section summarizes the key aspects of every model and visualizes them in the following tables. On the left side of every table, a brief description and some of the advantages and disadvantages of the according access control model are presented. On the right side, the applicability for the framework as additional access control model and some possible use cases are proposed.

Discretionary Access Control (DAC)

The access to system resources is controlled by its ownership, means that the access is controlled based on the identity of the user, who wants to access the resource.	Based on the identity of the intruder, the behavior of the framework may differ. Further the user is able to customize the access rights for specific resources.
+ Flexible, thus users are able to enable access to their own files for other users. + Easy to implement, e.g., as ACLs or authorization table. – Policy is not controlled centrally by an administrator and users are allowed to override access rights. – Changing or updating access rights is time-consuming, especially in large organizations.	• If the intruder has the same access rights for a resource as the user, the framework does not have to deny the access. • The user is able to customize the framework in respect of access rights for particular persons. • Based on the identity of the user and his or her access rights, he or she is allowed to customize the framework.

Table 2.3: Key aspects of DAC

Each discussed access control model has its own advantages and disadvantages and can be used differently for the framework *Confidential Desktop*, depending on what is to be achieved. In order to keep the proposed solution as flexible as possible and easy adaptable for various scenarios, all models except MAC are preferred. Of course a combination of several models can be used to increase security.

Mandatory Access Control (MAC)	
The access to a system resource is controlled by the operating system, based on MLS and different security levels.	Based on the assigned security level of an intruder or an object, the behavior of the framework may differ.
+ Thus only the system administrator is allowed to determine and change access rights, MAC offers high security. + Access policies can not be overwritten by any user, neither intentionally nor accidentally. – Complicated regarding configuration, because for every resource its security level has to be determined.	• Consideration of security level clearance of an entering intruder and the security level classification of the currently used resource. • Definition of a particular security level “threshold” for the framework, at which it is intended to regulate access. • Based on the clearance of the user, he or she is allowed to customize the framework.

Table 2.4: Key aspects of MAC

Role-based Access Control (RBAC)	
Access permissions are assigned to roles instead to individual users.	Based on the assigned role of an intruder, the behavior of the framework may differ.
+ Changing or updating access rights is less expensive in time and complexity, especially in large organizations. – Access permissions are static and do not consider contextual information about subject or object.	• A currently running application is closed by the framework, if an intruder with the role *Supervisor* enters the room. • The user is allowed to customize the framework, if he or she has the role *Admin.*

Table 2.5: Key aspects of RBAC

Attribute-based Access Control (ABAC)	
An access control model, which takes additional, predefined attributes of subjects, objects or environmental conditions into consideration.	An additional inclusion of characteristics of users and access controlled resources into the access control mechanism of the framework.
+ Flexible, because an unlimited number of attributes can be considered. + Dynamical access permissions, which consider contextual information. – High effort in the technical implementation. – Each ABAC policy must be specially adapted for a scenario.	• The access to a currently used resource is denied by the framework, if the cleaning staff enters the room. • The type of a resource is considered, e.g., the access to applications is permitted, the access to text files not. • Based on the location of the user (office or home), the behavior of the framework differs.

Table 2.6: Key aspects of ABAC

Context-aware Access Control	
An access control model, which considers environmental conditions and contextual information.	Additional inclusion of contextual information into the access control mechanism of the framework.
+ Flexible, dynamically access control mechanism (cf. ABAC). – High effort in the technical implementation, especially if specific sensors are needed for e.g., temperature measurement. – Can be quite complex to implement when, e.g., several external devices are integrated.	• Deny access to resources if the sound level in the office is high. • The behavior of the framework may differ based on the day of the week (work days, weekend, holidays). • Networking of multiple external devices (smartphones, tablets) to observe several areas.

Table 2.7: Key aspects of context-aware access control

CHAPTER 3

Background and Related Work

The purpose of this Chapter is to review the literature on computer vision and audio analysis, which is relevant for the proposed framework *Confidential Desktop*. It begins by explaining the principles of various methods in the area of computer vision, e.g. human or face detection, and discussing different approaches. This is followed by a summary of related work in the area of audio analysis, containing, among others, audio event detection or speaker recognition.

3.1 Computer Vision

This Section shows the large amount of opportunities offered by the field of computer vision to detect the presence of one or more persons in the room, e.g., via webcam. The following discussed approaches are a few examples, that could be used to extend the framework dynamically. These approaches, e.g. for human detection, are relevant for the framework, as they provide powerful methods for visual recognition of an entrance of an intruder.

3.1.1 Human Detection

Human and pedestrian detection is a well-known problem in computer vision with applications in surveillance [KWSN96], care for elderly [RFY$^+$13] or safety of driving [SGH04], and many more.

A common approach for human detection is the computation of Histogram of Oriented Gradients (HOG) descriptors. N. Dalal and B. Triggs [DT05] first used Histograms of Oriented Gradients in combination with a linear Support Vector Machine (SVM) classifier to detect pedestrians in static images.

HOG is a feature descriptor used in computer vision and image processing. It detects objects by the distribution of gradients in the image, where a gradient represents a directional change in image intensity or color. For example, humans in standing positions cause strong vertical edges along the boundaries of their bodies, thus HOG features are particularly suitable for human detection in images.

In the following, the general computation of an HOG descriptor is explained in more detail. For the sake of simplicity, normalization is not taken into consideration and can be found among others in the work of Dalal et al. [DT05].

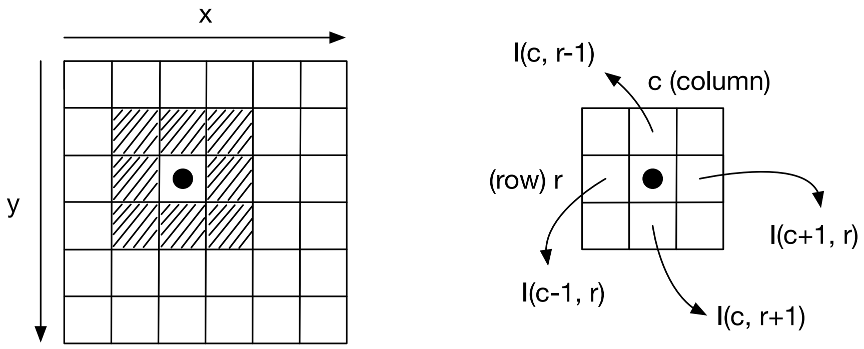


Figure 3.1: Neighboring pixels for gradient orientation and magnitude

To compute an HOG descriptor, first the image is divided into small cells and for every pixel within these cells, gradient orientation and magnitude are computed. This is done by subtracting the values of the neighboring pixels (Figure 3.1) to get dx (the change in x direction) and dy (change in y direction):

$$dx = I(c+1, r) - I(c-1, r) \tag{3.1}$$

$$dy = I(c, r-1) - I(c, r+1) \tag{3.2}$$

After that the gradient orientation $go = tan^{-1}(\frac{dy}{dx})$ and the magnitude $m = \sqrt{dy^2 + dx^2}$ can be computed.

The next step is the generation of a histogram for every cell. The amount of magnitude that is split up into separate histogram bins, depends on respective orientation of each gradient vector.

E.g., consider following example of a 9-bin histogram, 20 degrees per bin (Figure 3.2) and a gradient vector with an orientation of 85 degrees. $(90 - 85)/20 = 0,25$ means that 25% of the magnitude goes into the 4th bin and $(85 - 70)/20 = 0.75$ means the other 75% of the magnitude goes into the 5th bin. After all histograms were computed, their concatenation represents the HOG descriptor.

Based on the work of N. Dalal and B. Triggs [DT05], there are various extended approaches, for example by using Principal Components Analysis (PCA) to reduce

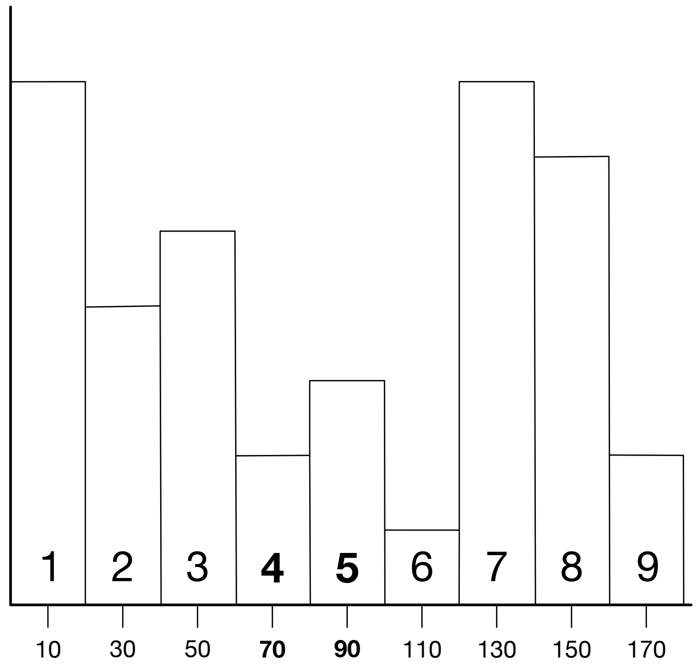


Figure 3.2: 9-bin histogram (0-180 degrees)

the dimensionality of the feature descriptors [KHK08], or by using Adaptive Boosting to speed up the computation [ZYCA06].

Another approach for human detection is the use of wavelet templates [Mal89, POP98, PP99], which “define the shape of an object in terms of a subset of the wavelet coefficients of the image” [OPS$^+$97].

While the above mentioned approaches describe holistic human detection, i.e., detecting people as a whole, there are also part-based [WN05, MSZ04] and patch-based [LSS05] pedestrian detection approaches.

One of the latest surveys, written in 2014 by Benenson et al. [BOHS14], compares the detection quality of over 40 different pedestrian detection methods by the use of the Caltech-USA dataset [DWSP09]. Further the authors conclude that “most of the progress in the last decade of pedestrian detection can be attributed to the improvement in features alone” and that ”this trend will continue”.

3.1.2 Face Detection

Face detection is like human detection, one of the most studied topics in computer vision and Human-Computer Interaction (HCI) and has therefore spawned a lot of different approaches for facial analysis algorithms in recent years. These approaches range from face recognition [TP$^+$91b, PMR$^+$00, ZCPR03] and authentication [BLGT06, DFB99] to head pose [YZ02, MCT09] and facial expression tracking [KCT00, FL03].

Further, nowadays face detection is increasingly used in everyday technologies, for example in biometric authentication [SKK04], digital photo management software [CHZ$^+$03], and cameras [RK08], which use face detection for autofocus. Zhao et al. [ZCPR03] mention in their work also advanced video surveillance, video games and TV parental control as typical applications of face recognition.

In 2002, Yang et al. [YKA02] group single image face detection techniques into four different categories, which are still referenced in current surveys: (i) knowledge-based methods, (ii) feature invariant approaches, (iii) template matching methods, and (iv) appearance-based methods. Table 3.1 summarizes the presented results and points out the differences among the four categories.

A more recent survey [ZZ10] gives an overview on various advances in the area of face detection. They go beyond the beginnings of face detection and show how advanced the mechanisms today already are. Their overview of features for face detection contains, among others, statistics-based features, as for example spectral histograms [WL$^+$05]. Further they also discuss the challenges of this research area, which are still faced, e.g. speed issues and multiview face detection.

A look at recent works shows, that approaches for face detection are becoming increasingly sophisticated. Kovač et al. [KPS03] for example discuss in their paper skin color based face detection. They further describe an approach for the use of an appropriate color space to deal with illumination conditions. Dantone et al. [DGFVG12] propose conditional regression forests to detect the location of feature points of faces, dependent on different head poses. The authors of [PPT$^+$11] are also dealing with head poses, by proposing a real-world 3D face recognition method that uses facial symmetry.

3.1.3 Background Subtraction

Background subtraction is a widely used strategy to detect moving objects in videos. In this technique, a so-called reference frame (the *background image*) is created and compared with the current frame to check whether there are any differences between them. This method is also called *frame differencing.* To improve the results of the background subtraction, a threshold can be used. If the difference between the pixel values P of two frames F at the time t is above a certain threshold, movement is

	general approach	**issues**	**examples**
knowledge-based methods	human knowledge about the appearance of human faces (relations between facial features) is defined as rules for face localization	difficult to translate human knowledge into well-defined rules, too strict rules lead to false negatives and vice versa	Hierarchical knowledge-based method [YH94]
feature invariant approaches	facial features are extracted (e.g. eyes, eyebrows, etc.) by using edge detectors and a statistical model is built for face localization	features can be corrupted by occlusions, noise and illumination (shadows)	Skin Color [MGR98] Multiple Features [KK96] Facial Features [LBP95]
template matching methods	models/templates for face patterns, which are used for face localization and detection, are predefined as rules or functions	dealing with variations in shape, pose and scale of faces	Predefined face templates [CTB92]
appearance-based methods	models/templates for face patterns, which are used for face detection, are learned from a set of training images	depend on how well the machine learning and statistical analysis algorithms work	Eigenface [TP91a] Support Vector Machine (SVM) [OFG97]

Table 3.1: The four categories of single image face detection techniques [YKA02]

detected:

$$|P\left[F\left(t\right)\right] - P\left[F\left(t+1\right)\right]| > Threshold \tag{3.3}$$

Using this method, the speed of the moving object in the video must be considered in order to obtain useful results. The higher the speed of the object, the higher the threshold should be.

Another method to obtain the background image $B(x,y,t)$ of a number N of frames is to calculate the arithmetic mean of them, whereby x and y are pixel location variables, t the time, and N dependent on the speed of the video (frames per second):

$$B(x,y,t) = \frac{1}{N}\sum_{i=1}^{N} F(x,y,t-i) \tag{3.4}$$

Horprasert et al. [HHD99] propose a real-time algorithm to detect moving objects in static background images. Their goal was to create a robust background subtraction algorithm even for scenes with illumination changes (e.g., shadows). This is achieved by using a color model, which separates the brightness of the colors from their chromaticity. The authors refer at this point to the human perception of color, i.e., people have the tendency to assign a constant color to an object, regardless of lighting conditions [Hur89].

Another approach, from Elgammal et al. [EHD00], deals with background subtraction with only partially static backgrounds. The authors describe a non-parametric background model which is able to handle outdoor scenes with little motions such as bushes and tree branches that move in the wind. Furthermore, their proposed model is able to adapt quickly to changes in the scene and works for both color and grayscale images.

Recent surveys provide an overview of common background subtraction techniques and algorithms [Pic04, BJE$^+$08]. They summarize how the algorithms work and compare them with each other with respect to their performance and accuracy. Methods, which are reviewed, are for example temporal median filter [LV01, CGPP03], One Gaussian (1-G) [WADP97], Gaussian Mixture Model (GMM) [SG99, ZZ05] and Kernel Density Estimation (KDE) which is used by the above mentioned approach of Elgammal et al. [EHD00]. It is reviewed, that the temporal median filter is the most basic method for background subtraction, since the background model is computed based on the median value of the last n frames. This method has the disadvantage that all the pixel values of these frames must be buffered. Nevertheless, Benezeth et al. [BJE$^+$08] come to the conclusion, that this method is well suited for videos with static, noise-free background and more complex methods do not achieve significant better results. In contrast, 1-G, GMM, and KDE are much more reliable when it comes to noisy video sequences.

3.1.4 Body Tracking

As already mentioned in 2010 by Sminchisescu and Triggs [ST01], extracting a 3D human body model and its motion from an monocular video sequence is a challenging task which requires an enhanced approach which takes following three difficulties into consideration:

- Even a minimal human body model is very complex, it contains at least 30 joint parameters, deformable body parts, etc.
- In monocular video sequences not all degrees of freedom are recognizable, i.e., movements in (relative) depth are difficult to detect
- Generally, in scenes with cluttered backgrounds and variable light conditions it is difficult to match a such complex model, which is hardly known and also could be self-occluding.

Of course, these problems mentioned are not independent but can also influence each other. Nevertheless the authors take all three difficulties into account and present a system for 3D human body tracking in monocular video sequences. They further use Covariance Scaled Sampling for optimization.

Another approach for 3D human body tracking based on PCA is discussed by Urtasun et al. [UFF06], who use activity-specific, learned motion models for e.g., walking or running. Further they show their results both with monocular and multi-view people tracking.

Porikli et al. [PT03] propose a real-time human tracking system and address various fundamental issues like speed, robustness, illumination changes, shadows, etc., which are also often faced by other approaches. The authors discuss and compare respective state-of-the-art methods, e.g., for background subtraction and object tracking and integrate them into a robust real-time human tracking system for high-resolution color videos. Finally this system is based on GMM for adaptive background models and mean-shift analysis for a forward-tracking mechanism. It also includes a change detection method, which means that the background model is adapted if a change in illumination is detected by the system.

In 2013, Han et al. [HSXS13] gave a review about computer vision topics using the Microsoft Kinect Sensor. Kinect, containing, among others, a 3D depth sensor, a RGB camera, an infrared camera, and respective software, is one of the best known tools which can be used for human body tracking. Especially the fact that it includes a depth sensor, the low-cost Kinect is very useful for computing robust background models even at illumination changes or poor contrast. The authors analyze different approaches for object tracking using Kinect and further investigate the combination of depth images and RGB images for a GMM-based background subtraction method.

As mentioned at the beginning of this Subsection, (3D) body tracking is an advanced method which can be used to track an already detected person in the room and check, if an intruder is coming closer to the workstation or the laptop screen. Especially for the proposed framework *Confidential Desktop*, this technique could be further improved by implementing an additional trajectory prediction algorithm [RS98], so that it can be detected early, in which direction the intruder will go. Since this Section provides an overview of some basic techniques of the field of computer vision, a detailed discussion of this research area would go beyond the scope of this thesis.

3.1.5 Gaze Estimation

Particularly, when it comes to the case that an intruder is not only in the room, but close to the workstation, the laptop screen could be in his or her field of vision. In another scenario, it could be common that one or more other persons stay in the room and it should be checked whether someone just passes the screen or has a look on it.

One of the prior approaches is from Rikert et al. [RJ98] and is based on morphable models. They first detect a face over a single image and then extract the eye region and further information about head orientation and iris positions, using the morphable model. This model was created based on example images of frontal faces and various corresponding head orientations. In the end, the authors are able to estimate the screen coordinates, the test users were looking at.

Another approach which also concentrates on single images is discussed by Wang et al. [WSV03]. Unlike the above method, however, this approach is limited to only one eye. As in the images only one eye is seen, the iris can be discovered on the basis of the contrast between the eyeball and eye white. After the iris is detected, the most important step is to fit its contour exactly into an ellipse. Finally, the gaze can be estimated based on the normal to the plane of the iris ellipse.

Zhu et al. [ZJ04] propose a real-time gaze estimation system based on Generalized Regression Neural Networks (GRNNs). Their gaze estimation algorithm consists of three steps: pupil and glint detection and tracking using IR illumination, gaze calibration via GRNNs and gaze mapping. The purpose of their gaze estimation system was to interactively control graphical content on screens. For example, if the user sees a map, he can look at a certain region and then blink three times. After that the chosen region is magnified to fill the screen again, i.e., the user is able to use his eyes to communicate with the computer and to zoom in multiple times until the region of interest is detailed enough.

Yoo et al. [YC05] point out that many approaches for gaze estimation, e.g. the above mentioned system of Zhu et al. [ZJ04], of course take head movements into consideration. But these systems often require the use of multiple devices or consuming calculations. As opposed to this, the authors propose a fast and reliable method for gaze estimation, even under large head movements. Their presented method is also based on light reflections, i.e., four IR LEDs, which are placed on the corners of the monitor, are used to produce glints in the user's eyes. This means that the rectangle, which is encompassed by the four glints in the eye, represents the reflection of the screen. These eye glints are detected by a robust feature extraction method which includes, among other steps, the fitting of the iris contour into an ellipse, as already used in the approach of Wang et al. [WSV03]. Finally, a cross-ratio was used to estimate the eye gaze point.

Another approach from Valenti et al. [VSSG09] combines a state-of-the-art method for detecting the eye center with a new method for detecting the eye corners. For their proposed system they are only using a webcam to estimate the gaze of a user sitting in front of the screen. The authors argue that detecting only the location of the eye center is not enough for estimating the visual gaze of a person, an additional "anchor point" is needed for reliable results. The accuracy of the results of their proposed method is also dependent on the quality of the used webcam but therefore a light and flexible solution for approximate gaze estimation.

In a later work, Valenti et al. [VSG12] focus on the combination of head poses and eye locations. The result is a sophisticated, automatically self-correcting system which integrates head pose estimation based on Cylindrical Head Models (CHMs) and an isophote-based eye location estimation.

As this Subsection has shown, there are very different approaches both to implement and to use gaze estimation. Depending on how this technique is applied, different objectives can be achieved, ranging from the mere recognition of the head pose and tracking of eyes, to detecting whether a person roughly is looking in the direction of the screen or even determining the almost exact position of the user's gaze.

3.2 Audio Analysis

The last Section has demonstrated that there are many powerful methods in the area of computer vision and image processing, e.g., human, face or motion detection, which fit into the framework as features. As already mentioned in the introduction of this thesis, also the field audio analysis provides various methods to detect an intruder in a room. Using the audio analysis, a lot of different features can be implemented, ranging from simple noise detection or audio classification to more sophisticated techniques such as voice activity detection and speech recognition. In the following the most relevant methods are discussed in detail.

3.2.1 Audio Event Detection

In contrast to the rather simple noise detection, which detects only whether the volume does not exceed a certain value, the audio event detection is able to differentiate among different types of noise and differentiates between different events.

This concept is described in detail by Xiong et al. [XRDH03], who used audio event detection to detect highlights in baseball, golf and soccer games. By using Moving Picture Experts Group (MPEG)-7 audio features and Entropic Prior Hidden Markov Models (HMM) the authors extract and classify audio events like applause, ball-hits or cheering from noisy sport audio tracks. Figure 3.3 shows an overview of their proposed algorithm consisting of the three steps: background noise recognition, audio classification and post-processing.

Clavel et al. [CER05] use audio event detection to automatically detect abnormal sounds like gun shots in audio-based surveillance systems. They argue that in some situations, especially in the case of a shooting incident, an audio-based surveillance system can be more effective than a purely visual system. Based on the used audio information, the authors can not only detect when a gun shot occurs, but they can even distinguish between different types of weapons like grenades, submachine guns or rifles.

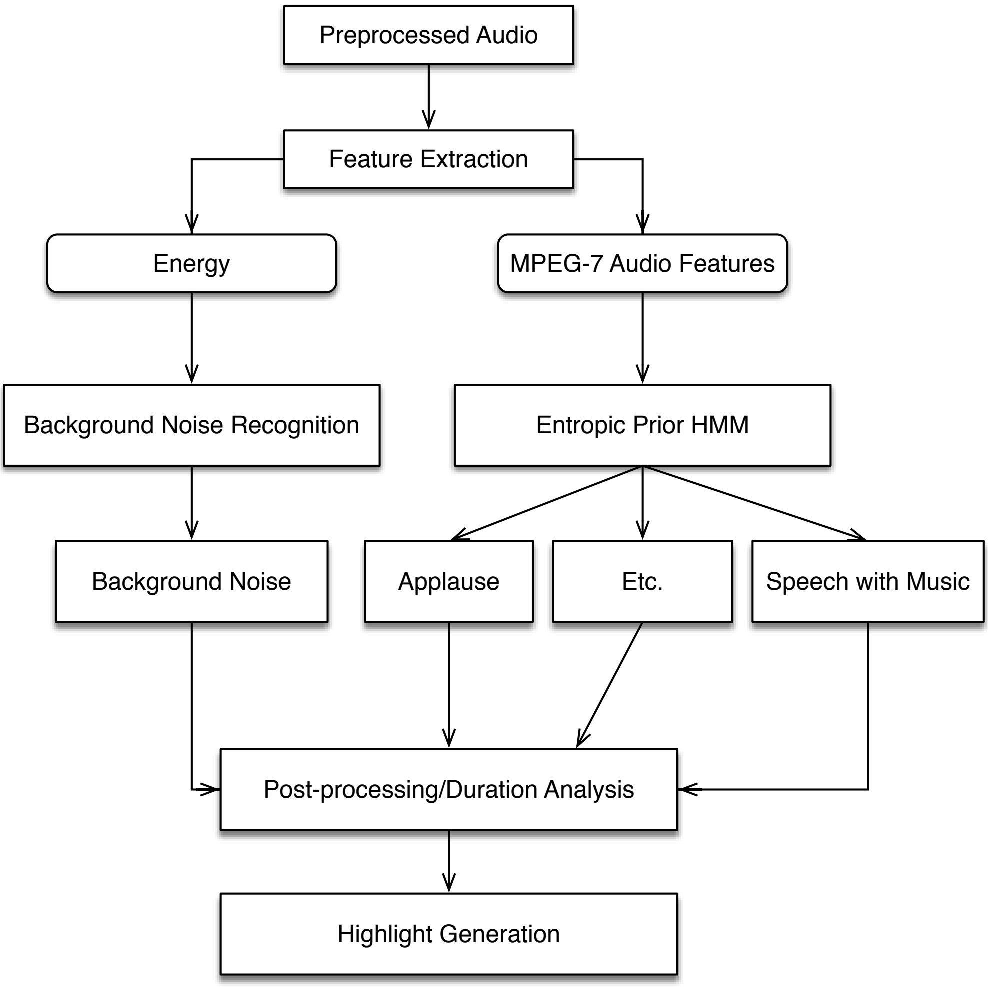


Figure 3.3: Algorithm Flowchart for Audio Event Detection [XRDH03]

Another paper from Portêlo et al. [EBT+09] deals with audio event detection in audio tracks which do not contain any speech. Also by using HMM in combination with a SVM, the authors detect a wide range of audio events, for example cat meowing, dog barking, water or traffic.

Last but not least Giannoulis et al. [GSB+13, GBS+13a, GBS+13b], proposed an evaluation challenge in 2013 for detecting and classifying acoustic scenes (e.g., office, restaurant, park) and events (e.g. door knocking, printer, speech), including datasets. Subsequently, several submitted works were evaluated and their results compared.

3.2.2 Voice Activity Detection

Voice Activity Detection (VAD) is concerned, as the name suggests, with the detection of human speech in audio, whereby the distinction from non-speech sounds plays a major role. Ramírez et al. [RGS07] describe the fundamental process of VAD.After the background noise reduction, speech features are extracted, which can be used for decision rules. These rules often are enhanced with smoothing algorithms to improve the robustness of the speech detection against noise.

Further the authors point out typical application areas where VAD is used, namely applications in which a reduction of background noise is important to hear voices better, for example in mobile communication technology [FCSB89] or for digital hearing aid devices [IM97].

Although VAD is a relatively young field of research, there are already different approaches, which are mentioned by Ramírez et al. [RGS07]. For example a VAD algorithm based on the Likelihood Ratio Test (LRT) in combination with a Discrete Fourier Transform (DFT) to obtain decision rules and an HMM-based hang-over scheme for decision smoothing [SKS99].

An earlier publication by Ramírez et al. [RSB$^+$04] describes an approach for VAD based on estimating the Long-Term Spectral Envelope (LTSE) and computing the Long-Term Spectral Divergence (LTSD) between speech and noise to define appropriate decision rules. Further they only use a hang-over scheme when the Signal-to-Noise Ratio (SNR) is low.

One of the latest approaches originates from Ma et al. [MN13], who propose a VAD algorithm using Long-Term Spectral Flatness Measure (LSFM), which can be also used for lower SNR scenarios, Gao et al. [GCZ$^+$13] describe an algorithm based on a Probability Distribution Function (PDF) of Fast Fourier Transform (FFT) magnitudes.

3.2.3 Speaker Recognition

While the preceding Subsection was concerned with the discovery of the activity of human speech in audio, the challenge now is to identify who is speaking. Speaker Recognition or also called “Voice Recognition” is an important technology used e.g., for biometric authentication.

Generally, a distinction is made between speaker identification and speaker verification. Joseph P. Campbell, Jr. [CJ97] discusses in his work, among other things, these two different approaches (see Figure 3.4). In Automatic Speaker Verification (ASV), the system verifies the identity claimed by a speaker by his or her voice (see “text-independent speaker recognition” below) [RQD00, ACLT00], or by a predefined phrase like a PIN or password (see “text-dependent speaker recognition” below) [Bur87, Far95]. In Automatic Speaker Identification (ASI) the system derives based on a given voice, who the speaker is and identifies, if the voice is known to the system [RR$^+$90, RR$^+$95].

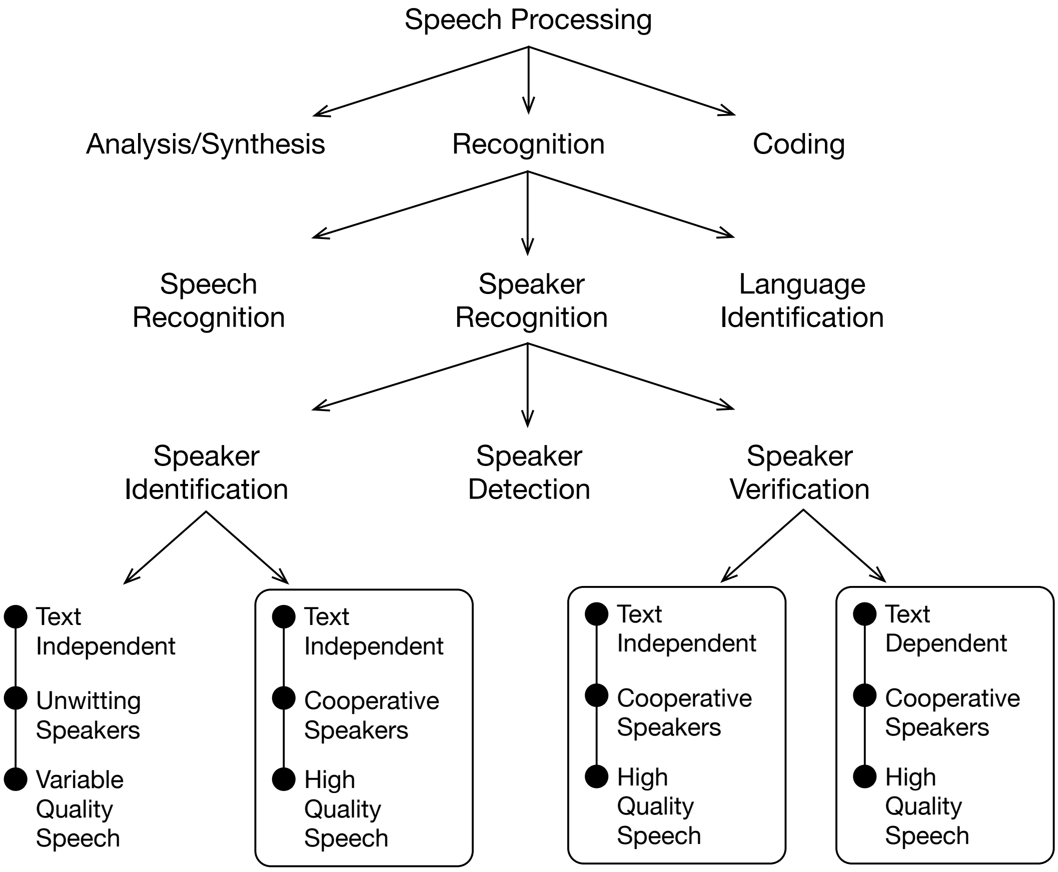


Figure 3.4: ASI vs. ASV [CJ97]

As already mentioned above, speaker recognition can be mainly divided into two variants:

- **Text-Dependent Speaker Recognition** When the speaker's voice is recorded for feature extraction, the recorded text has to be the same as the text which is also verified. Further this variant can be used in combination with knowledge-based information like e.g., a password, to establish a multi-factor authentication.

- **Text-Independent Speaker Recognition** In contrast to the text-dependent speaker recognition, the recorded text and the verified text need not be the same. This also means that no cooperation with the user is required and it is often implemented in combination with a speech recognition. This means that it is not only verified who is speaking, but also what he or she is saying.

Kinnunen et al. [KL10] provide an overview of text-independent speaker recognition and discuss, among other things, classical approaches for text-independent speaker models, e.g. based on GMM [RQD00, $\text{RR}^{+}95$], SVM [$\text{CCR}^{+}03$] or the combination of both [$\text{CCR}^{+}06$, CSR06]. Further they discuss recent research problems, challenges and trends regarding the field of speaker recognition.

CHAPTER 4

System Design and Implementation

So far this thesis has focussed on the theoretical part, the previous chapters show the wealth of opportunities to equip the framework with observation features, which observe the environment continuously. Of course, these are only a small part of the actually existing possibilities. By connecting other mobile devices and by the growing availability of different sensors, the framework may be adapted to different scenarios and dynamically expanded at any time (see also Section 6.2).

The practical part of this thesis is divided into the following parts:

- **Implementation of the framework** The proposed framework *Confidential Desktop* is implemented as a cross-platform desktop application in Java (Section 4.1), including a graphical user interface (Section 4.2) and the Intruder Activity Manager (Section 4.3).
- **Features extraction** Five exemplary background features calculated over the sensor firings are implemented, which continuously check the environment for bluetooth devices, faces, motion or noise (Section 4.4). At this point, the activity of an intruder is predicted based on sensor firing observations.
- **Performance of scenarios** Three different simulated scenarios are performed, where the user is sitting at his or her workstation and an intruder is entering the room a few times (Section 5.1).
- **Evaluation** The results from the three previously performed scenarios are evaluated. The detection quality of each background feature in every scenario and further found dependencies or tendencies are presented and discussed (Section 5.4).

The purpose of the framework *Confidential Desktop* is to integrate an ubiquitous, context-aware access control mechanism into the working environment of an user. The functionality of the proposed framework is inspired by the approach of Ayers et al. [AS01], who describe a system for detecting various human actions like Sitting, Standing, Talking on Phone, etc. in a room from a taken video sequence. *Confidential Desktop* goes a step further and integrates different sensors (webcam, microphone, bluetooth, etc.) for detecting an intruder in a room in real-time. The framework limits the access to system resources depending on an intruder's actions. Figure 4.1 shows that depending on intruder's actions, access to system resources or applications is limited. These intruder's actions are detected by the features calculated over the sensor firing observations.

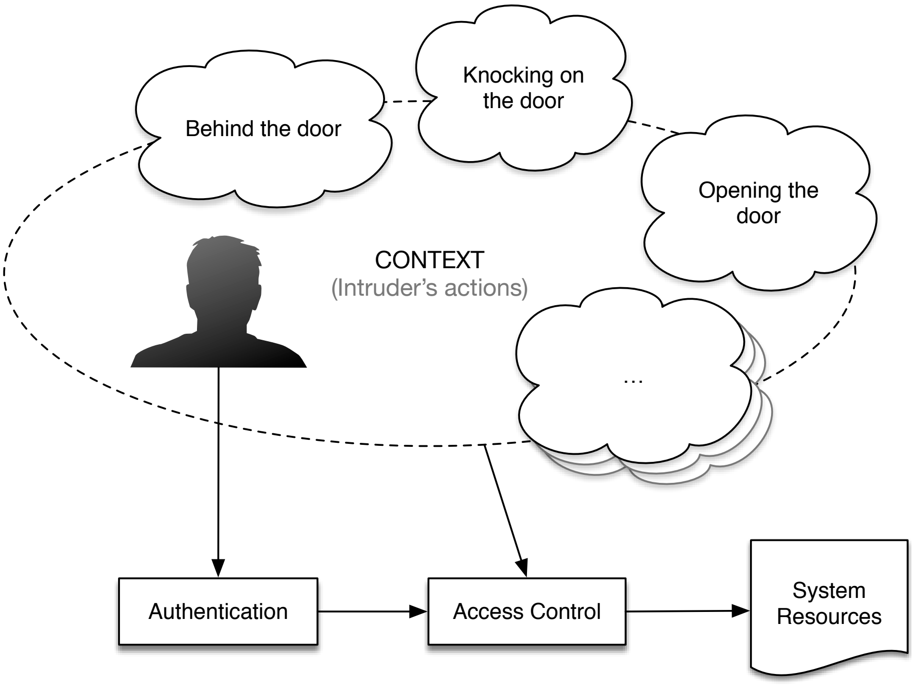


Figure 4.1: Context-Aware Access Control

The fact that the framework continuously checks a potential intruder's action and thus automatically controls access to applications, the confidentiality of security-critical data can be protected. If an intruder comes into the room or near the work station, this is recognized by the framework and even the authenticated user no longer has access to critical information. This prevents unauthorized persons of obtaining access to confidential data, whether accidentally or intentionally, e.g., by social engineering [Tho04, Wor07].

Figure 4.2 shows the conceptual structure of the *Confidential Desktop* framework. In the background, methods for recognizing intruder activities (e.g., Behind the door or Knocking on the door) are adapted to fit as features into the framework. The activity of an intruder is predicted based on features for bluetooth device discovery, face detection, motion detection and noise detection, calculated over the sensor (bluetooth sensor, internal/external webcam, smartphone camera, microphone) firings.

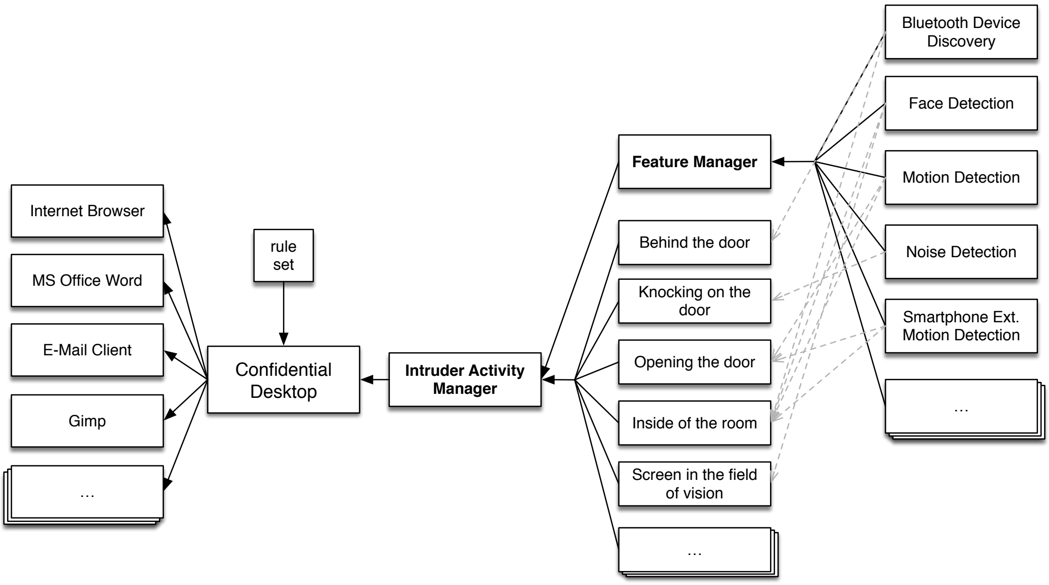


Figure 4.2: Concept of the *Confidential Desktop* Framework

All the background features, their current status, i.e., if they are enabled, and their current output, i.e., if they detect a threat for the user's data privacy, are managed by the `Feature Manager`. Various possible intruder's actions are managed by the `Intruder Activity Manager`, which is described in detail in Section 4.3. In the foreground the framework manages the accessibility of the applications, i.e., if the `Intruder Activity Manager` decides, that an intruder poses a threat to the user's privacy, the framework closes the predefined applications. At this point an additional rule set helps to define exactly how the framework should respond to the intruder's activities. For example, if the activity *Behind the door* is detected, nothing should happen at all. If the activity *Knocking on the door* is detected, the framework should minimize the Internet Browser window. If the activity *Screen in the field of vision* is detected, the framework closes the E-Mail client. After it is detected that the threat is not available any more, it reopens the applications again.

In the following Section, the system architecture of the framework *Confidential Desktop* is presented in detail.

4.1 System Architecture

The framework, started from its main component `ConfidentialDesktop`, first initiates all features and the main window of the graphical user interface. The next step is the loading of the last status of every feature from the `FeaturePreferences` and restoring the status of the last run, i.e., restarting all activity sensors which were running then (Listing 7.1). To understand all the activities of the framework and all events that have occurred during a session in retrospect, all data is recorded and stored in `FileWriter` (Listing 7.2) to analyse all activities in retrospect. Also all webcam video frames used for face or motion detection are stored into a folder for every session (Listing 7.3).

Since *Confidential Desktop* controls the access to various predefined applications, it has the ability to manage the currently running processes. The applications can be opened manually through the graphical user interface of the framework (Section 4.2). For every application a new process is created by the `ProcessBuilder` and further a `ProcessExitDetector` is assigned to listen, if the user is manually closing the application (Listing 7.4). This is required to maintain a representative list of all currently running applications.

Figure 4.5 shows the functionality of the framework. Every feature needs a specific input as stated in the figure. In case of face and motion detection for example, the features require images from the webcam in real-time, i.e., each frame of the webcam video. If a feature detects any relevant activity, e.g., the Noise Detection Feature detects a noise above a specified threshold, it sends a signal to the `Feature Manager`. The `Feature Manager` aggregates all reports from all features and forwards them to the `Intruder Activity Manager`, which decides whether the framework should grant access to the applications (because the user is alone in the room or not) or the framework should deny access and therefore close all open confidential applications (because an intruder's activity is detected). Further the framework knows at any time, which application processes are currently running, due to the monitoring capabilities of *Confidential Desktop*.

A far-reaching design decision regarding the architecture was how the `Feature Manager` gets the signals from the features. One possibility would have been to use a long-polling approach (Figure 4.3), i.e., the `Feature Manager` fetches periodically the current status from all existing features. The synchronization would therefore take place in the `Feature Manager`. The advantage of this approach is that there is always a consistent status of all features and thus dependencies between features can be easily implemented. Furthermore the performance is depending on the slowest feature, therefore this approach scales in a horizontal way assuming that the new added features are not slower than the existing ones. Thus the performance of the system heavily depends on the slowest feature and not on the amount of features. In context of scalability most of the decision process depends on the feature, therefore those components are most relevant. The actual calculation of the result is done by simple mathematical operations and due to its complexity

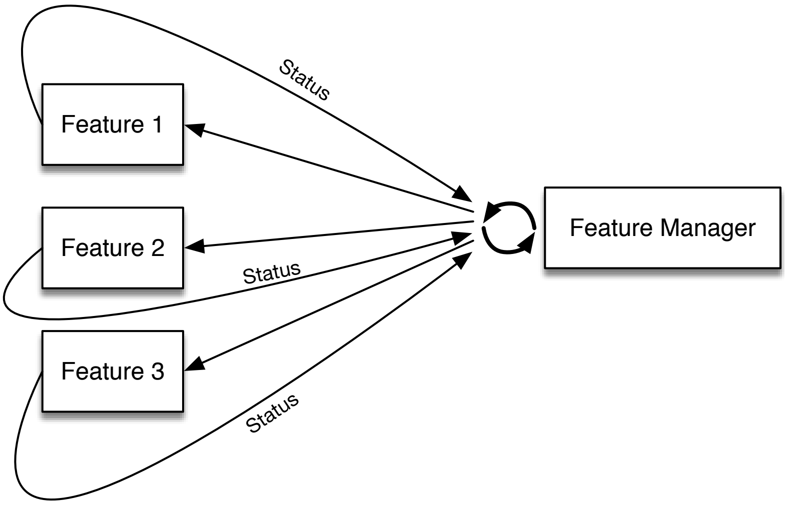


Figure 4.3: Long-Polling Approach

compared to the feature, it adds practically no weight to overall process and can therefore be neglected for scalability analysis. One disadvantage here is that the status of the first features already might have changed till the status query of the last feature is finished. In addition, rare updates are processed, since the `Feature Manager` always goes through all features and only then sends a collected status to the `Intruder Activity Manager`.

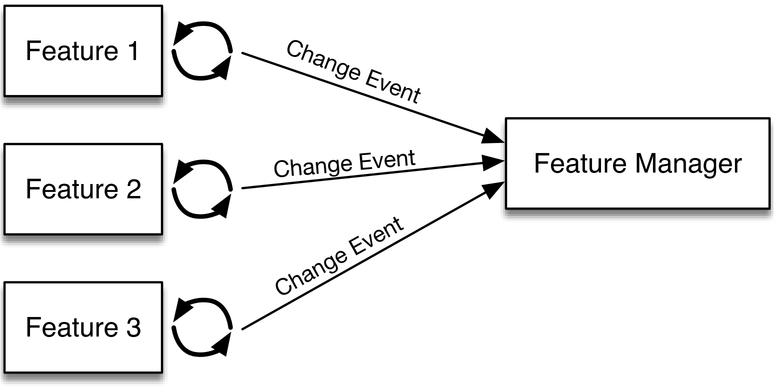


Figure 4.4: Event-Based Approach

Another approach is to implement an event-based (push) architecture (Figure 4.4). Synchronization here is done within the features, i.e., each feature periodically sends a new status event to the `Feature Manager` if its status has changed. This, in contrast to the above mentioned approach, has the advantage that the time between the action happened and the `Feature Manager` is informed is minimal and slow features do not block the query. Due to the faster update propagation, this approach was selected, although it may render an inconsistent state due to the not synchronized communication.

The focus for the implementation of *Confidential Desktop* was on platform independence, high flexibility and modular expandability. Java 8[1] was chosen as programming language for the framework thus it is object-oriented and platform independent. As operating system for implementation, Windows 8.1 (64-bit) was chosen. For detailed information about setup and used technologies, see Appendix 7.2.

---

[1] http://www.oracle.com/technetwork/java/index.html [Online; accessed 07.09.2015]

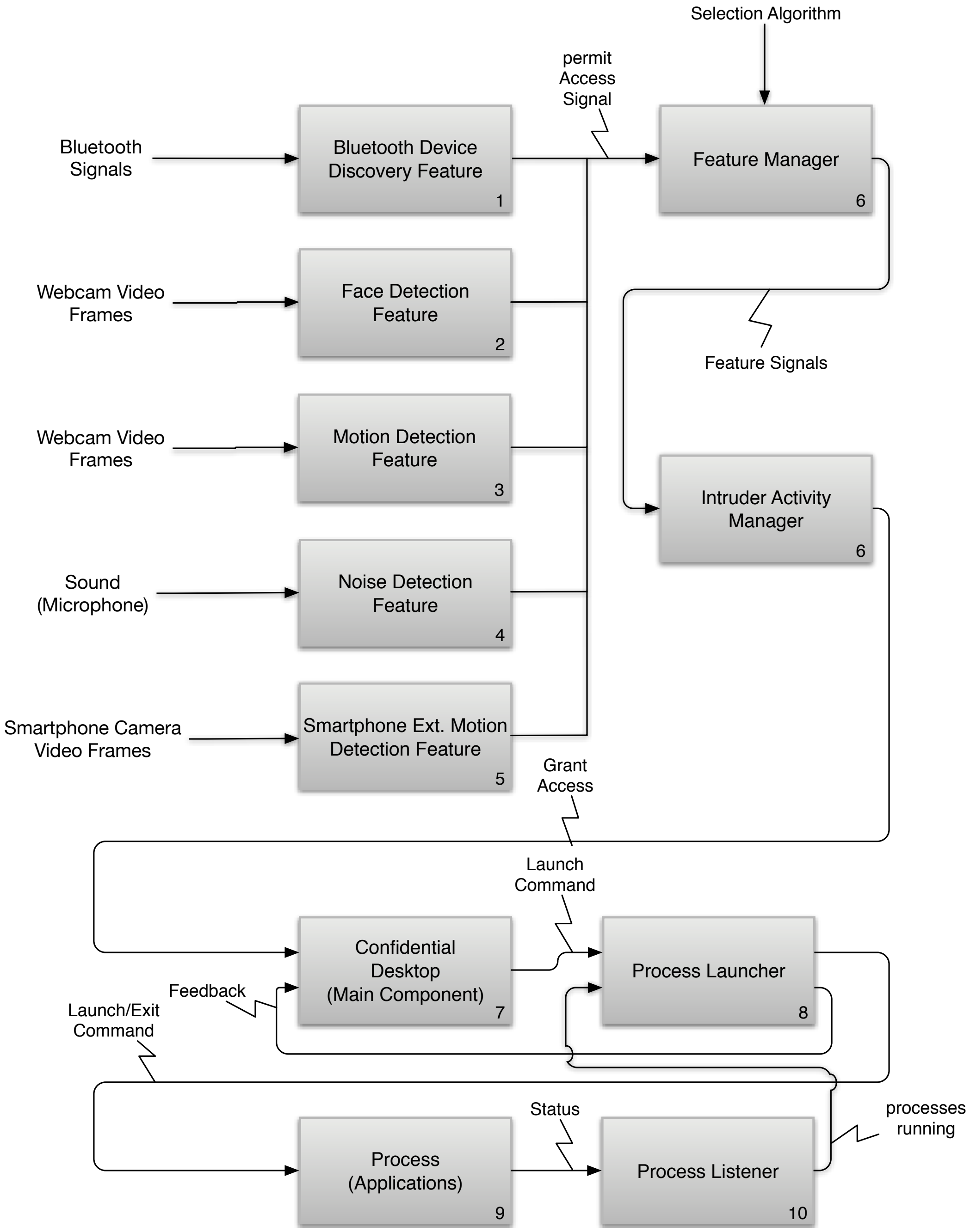
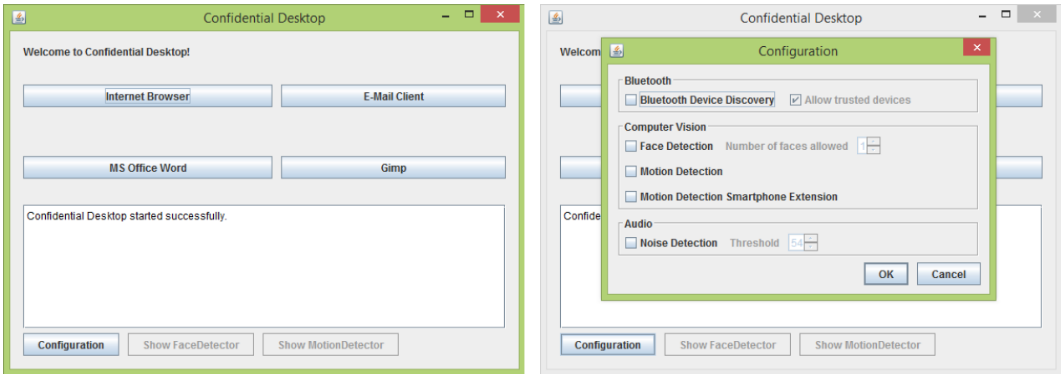


Figure 4.5: Process Overview (IDEF0 Model)

4.2 Graphical User Interface

The Graphical User Interface (GUI) of the framework *Confidential Desktop* is based on the Java GUI widget toolkit *Swing*[2]. With this toolkit, it is possible to quickly create a lightweight user interface.

The user interface allows the user to use and to adapt the framework to different scenarios. Figure 4.6 shows the main window and the configuration window. In the main window the user can manually access the applications which are controlled by *Confidential Desktop* by clicking on the dedicated button. This main window also shows the log output of the framework. At the bottom of the window there are three buttons. The left button is always enabled and opens the configuration window. The other two buttons are used to open the FaceDetector or the MotionDetector. They display the webcam video input in real-time including the detected faces or motion (see Figure 4.7) and are only enabled when the corresponding features are enabled.



Confidential Desktop started successfully.

☐ Bluetooth Device Discovery ☑ Allow trusted devices
☐ Face Detection Number of faces allowed 1
☐ Motion Detection
☐ Motion Detection Smartphone Extension
☐ Noise Detection Threshold 54

Figure 4.6: GUI Main Window and Configuration Window

Through the configuration window, the user can see, which features are available for the framework and can enable or disable each of them. Further for some of the features, settings are available, which can be adapted in the configuration interface. For example, for the face detection feature, it can be defined, how many faces should be allowed. The feature should only be triggered if two or more faces are in the observation area due to the fact that at least one person is working on the PC. If an external webcam is used, which points to an entrance door of a room, no face should be allowed and the feature should deny the access if one or more faces are detected. These individual settings also can not be accessed, if the corresponding feature is disabled, so first if a feature is enabled by the user, also its setting parameter can be

---

[2] https://docs.oracle.com/javase/8/docs/technotes/guides/swing/index.html [Online; accessed 07.09.2015]

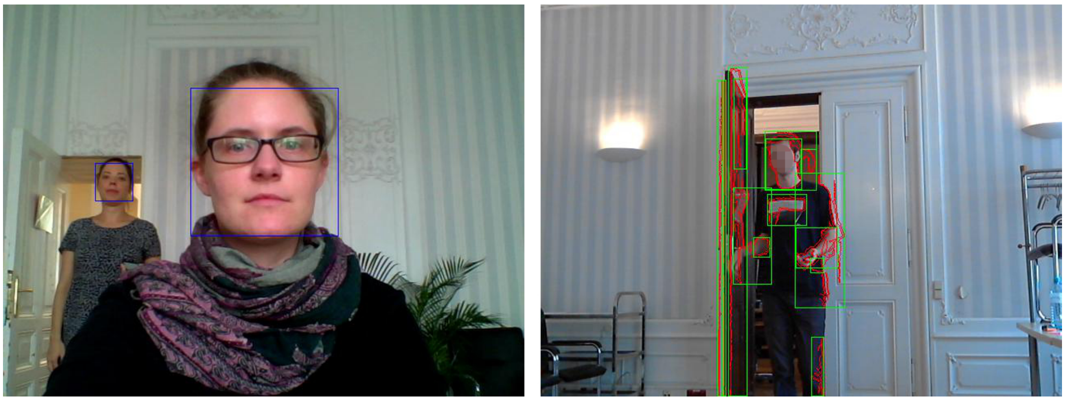
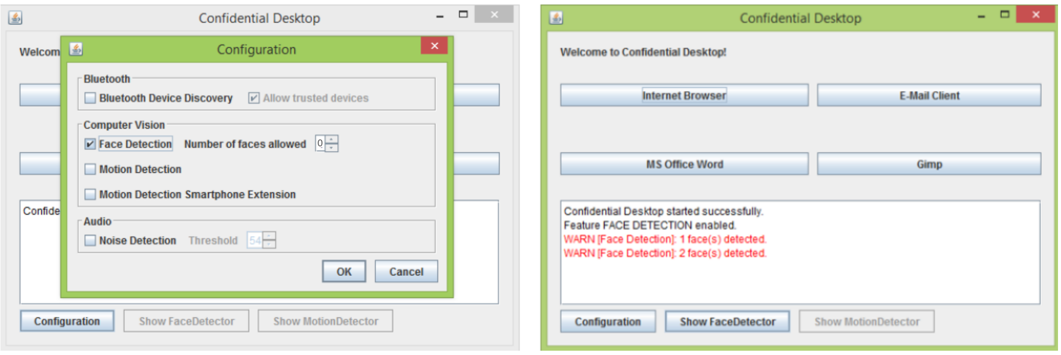


Figure 4.7: FaceDetection (Scenario 1) and MotionDetection (Scenario 3)

changed.

Figure 4.8 now shows the specific use case of enabling a feature. If the user enables the face detection feature, he or she can define the number of faces, which are allowed for the feature. After clicking the *OK* button, the configuration is saved (see also Section 4.4) and the face detection is enabled. The logging interface gives feedback to the user that this feature is now enabled and also prints warnings in red text color, if the feature detects more faces than allowed.



Bluetooth
☐ Bluetooth Device Discovery ☑ Allow trusted devices

Computer Vision
☑ Face Detection Number of faces allowed 0
☐ Motion Detection
☐ Motion Detection Smartphone Extension

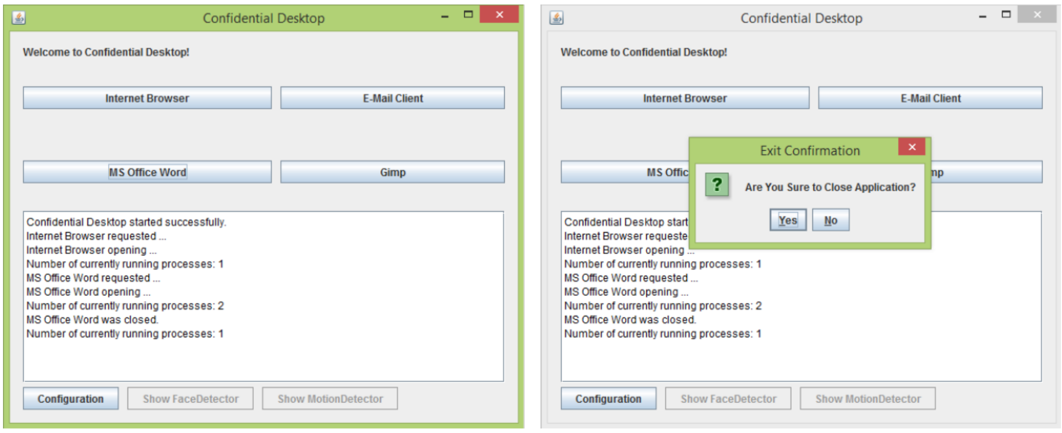
Audio
☐ Noise Detection Threshold 54

```
Confidential Desktop started successfully.
Feature FACE DETECTION enabled.
WARN [Face Detection]: 1 face(s) detected.
WARN [Face Detection]: 2 face(s) detected.
```

Figure 4.8: GUI Enabling Face Detection Feature

As already mentioned, the framework *Confidential Desktop* knows at any time, which and how many applications are currently running. If the user manually starts an application from the framework main window, he or she gets feedback from the framework in the form of a logging interface (Figure 4.9). Right after the click on one of the four application buttons the respective application is requested. If the corresponding process could be started successfully (Listing 7.4) the framework notifies the user, that the application is running. Further after a new application is opened or an already running application is closed by the user, the number of the currently running processes is printed by the framework, as can be seen in Figure 4.9 on the left hand side.

Finally, an exit confirmation dialog window was implemented, to prevent unintended closing of the framework. If the user triggers the closing operation provided by the operating system in the upper right corner of the window, a confirmation dialog opens and the user has to confirm, that he or she wants to close the application.



```
Confidential Desktop started successfully.
Internet Browser requested ...
Internet Browser opening ...
Number of currently running processes: 1
MS Office Word requested ...
MS Office Word opening ...
Number of currently running processes: 2
MS Office Word was closed.
Number of currently running processes: 1
```

Are You Sure to Close Application?

```
Confidential Desktop start
Internet Browser requeste
Internet Browser opening ...
Number of currently running processes: 1
MS Office Word requested ...
MS Office Word opening ...
Number of currently running processes: 2
MS Office Word was closed.
Number of currently running processes: 1
```

Figure 4.9: GUI Console Log Messages and Exit Confirmation Dialog

4.3 Intruder Activity Manager

The `Intruder Activity Manager` receives the collected signals of all features from the `Feature Manager` as input. The `Intruder Activity Manager` maps these input signals to predefined intruder activities. This component applies predefined rule sets, which are composed of a list of possible intruder activities and associated features which could indicate these activities. Table 4.3 shows an abstract rule set. If Feature 2 and 3 discover an activity and Feature 1 does not, this is an indicator for the Intruder Activity 1, and so on. The rule sets which are used for the evaluation of this framework, including all used intruder activities and features, can be found in Chapter 5.

The framework allows blacklisting as well as whitelisting activities. Blacklisting pursues an optimistic approach, where all activities are possible as long they are not stated explicitly. In other words, the access to all system resources is permitted,

Intruder Activity 1:	$\neg$ Feature 1 $\wedge$ Feature 2 $\wedge$ Feature 3
Intruder Activity 2:	$\neg$ Feature 1 $\wedge$ $\neg$ Feature 2 $\wedge$ Feature 3
Intruder Activity 3:	Feature 1 $\wedge$ Feature 2 $\wedge$ $\neg$ Feature 3

Table 4.1: Example rule set for Intruder Activities

unless the framework detects any intruder's activity.

Whitelisting on the other hand pursues a pessimistic approach, where only explicitly mentioned activities are possible. Means that access to system resources is prohibited unless there are, e.g., two user in front of the computer ("four-eyes principle").

For every system resource or application it can be defined, how the access should be controlled, i.e., blacklisting and whitelisting can be also combined.

4.4 Features

Figure 4.10 shows, all implemented background features inherit from the abstract `AbstractFeature` class, which has four methods. `start()` and `stop()` as their names imply are for starting and stopping the activity of a feature. If the user enables the feature by means of the configuration interface (see Section 4.2), the `start()` method is called. If the user disables the feature, the `stop()` method is called. The `isEnabled()` method returns, whether the specific feature is currently enabled or not. By calling the `permitAccess()` method it can be determined, if the feature is currently denying the access to any application, because it detected a second person in the room or not.

Thus all features inherit from this abstract class, all interfaces of the features are uniform and the framework can easily be extended by new features. All features are managed and accessed through the above mentioned methods by the `FeatureManager`. The `FeatureManager` knows at any time which features are currently activated and if they are currently detecting an activity. Each feature is self contained and does not interfere with the other features.

Additionally a `FeaturePreferences` class allows to persist the configuration settings of the features, i.e., it stores the configuration data and the current status of the framework beyond a shutdown.

The following sections will discuss the exemplarily implemented background features in detail.

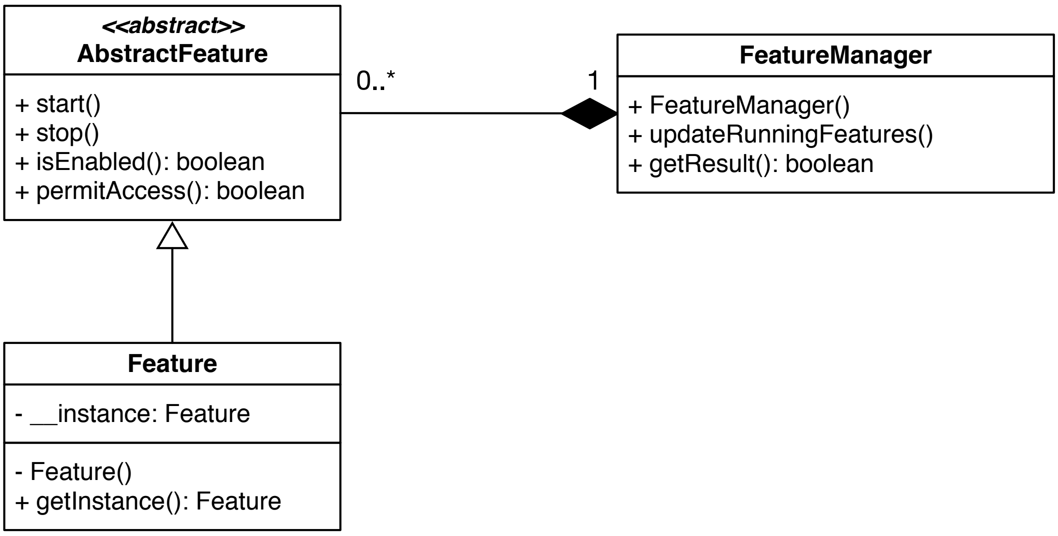


Figure 4.10: Class Diagram Features and FeatureManager

4.4.1 Bluetooth Device Discovery

For the Bluetooth Device Discovery feature Bluecove 2.1.1[3] and Quartz 2.2.1[4] were used for implementation. The feature uses a Quartz Scheduler to broadcast every few seconds a search for bluetooth devices in the close vicinity of the laptop. The feature can discover one or more devices at the same time, provided that they have bluetooth activated.

Furthermore, the device discovery is able to distinguish between a trusted device and an untrusted device. A trusted device was paired with the laptop previously (or is still), an untrusted device has never been paired with the laptop. The user can specify in the settings whether the feature should also consider trusted devices as problematic.

Listing 7.5 shows the quartz scheduler, which starts at activation of the feature and which performs the device discovery (Listing 7.6) every 20 seconds. If a device is found, the feature sends a signal to the framework, as an intruder with this device is located in the immediate vicinity.

---

[3] http://bluecove.org [Online; accessed 22.09.2015]

[4] http://quartz-scheduler.org [Online; accessed 22.09.2015]

4.4.2 Face Detection

With OpenCV for Java[5] the real-time face detection was implemented (Listing 7.7) to detect one or more faces simultaneously. OpenCV uses Haar Cascades [VJ01, WF06] for face detection. From positive and negative images (images with and without faces) haar features are extracted (Figure 4.11) and by subtracting the sum of pixels under the white rectangle(s) from the sum of pixels under the black rectangle(s) a single value is obtained to train the classifier.

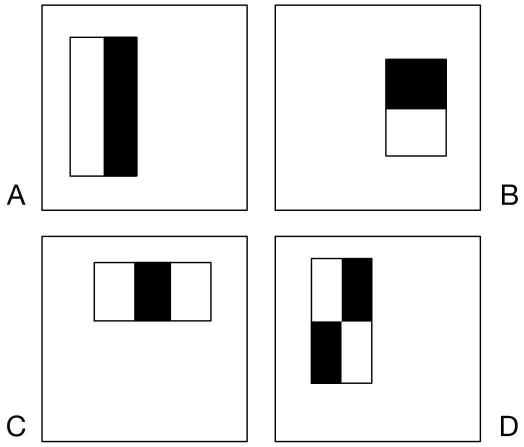


Figure 4.11: Two-rectangle features (A,B), a three-rectangle feature (C), and a four-rectangle feature (D) [VJ01]

When faces are detected, they are framed in a blue rectangle (Figure 5.5). By using the haar-cascade detection in OpenCV it is possible to use the already pre-trained classifiers for full/upper/lower body, frontal/profile face, eyes, etc. or to train the cascade classifier for any other object.

4.4.3 Motion Detection

The feature for motion detection is also created using OpenCV for Java. Once something or somebody moves in the view of the camera, e.g., when a door opens and an intruder enters the room, the movement is detected and the feature issues a warning. This is done by background subtraction (which is explained in Chapter 3, Subsection 3.1.3).

---

[5] http://opencv.org [Online; accessed 16.09.2015]

4.4.4 Noise Detection

For noise detection, the Root Mean Square Fluctuation (MSF) (root mean square fluctuation) is determined. The MSF is calculated for a timeframe T as follows:

$$x_{MSF} = \sqrt{\frac{1}{T}\sum_{t_j=1}^{T}(x_i(t_j) - \tilde{x}_i)^2} \tag{4.1}$$

This formula is implemented for the feature as follows:

Algorithm 4.1: Calculate MSF

```
Input: audioData (byte[] array)
Output: double
$sum = 0.0$;
for $i \leftarrow 0$ to $audioData.length$ do
    $sum += audioData[i]$
end
$avg = sum/audioData.length$;
$sumMeanSquare = 0.0$;
for $i \leftarrow 0$ to $audioData.length$ do
    $sumMeanSquare += (audioData[i] - avg)^2$;
end
$averageMeanSquare = sumMeanSquare/audioData.length$;
return $sqrt(averageMeanSquare)$;
```

The resulting value indicates the strength of the change of the volume over a certain period T. By using the Java Speech Application Programming Interface (API)[6], the noise detection feature receives the input from the internal microphone of the laptop in real-time. The user is able to set a threshold (the calculated MSF) manually via the GUI, which must not be exceeded. Once the threshold is exceeded, the feature sends a signal.

4.4.5 Smartphone Extension: Motion Detection

For this feature a smartphone Android application was implemented. By using the smartphone extension, it is possible to observe an additional area of the room. Thereby, the smartphone feature communicates with the framework and simply sends an event if it has discovered something (Figure 4.12).

For the Android application, one or more features can be implemented, which simply

---

[6] https://docs.oracle.com/cd/E17802_01/products/products/java-media/speech/forDevelopers/jsapi-doc/ [Online; accessed 13.11.2015]

need to send events to the framework. In this case a simple algorithm for motion detection was implemented.

This feature is perfectly suited to place the smartphone in the room with the camera pointing at a door to see if it opens and someone comes in.

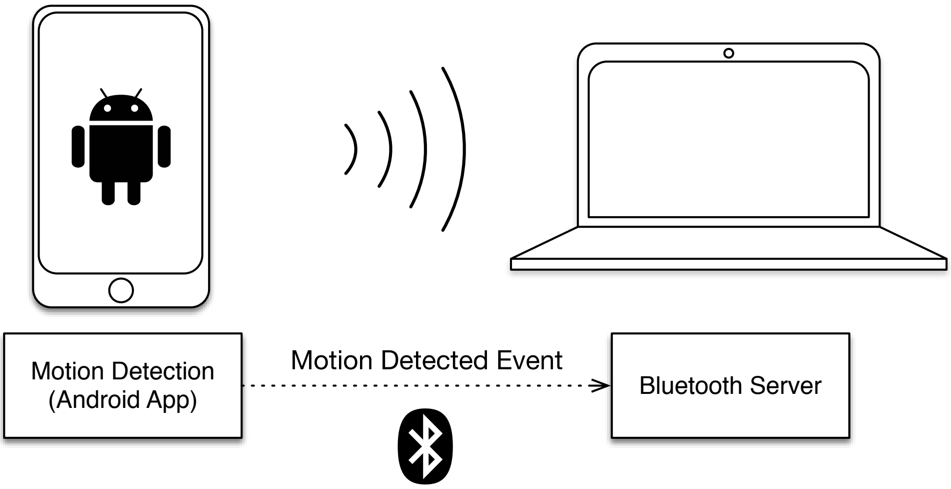


Figure 4.12: Smartphone Extension Motion Detection Feature

CHAPTER 5

Evaluation

In addition to the design and implementation of the proposed framework *Confidential Desktop*, the evaluation was an important practical part of this thesis. As already mentioned, for the evaluation three different scenarios were simulated and performed. This chapter describes the detailed structure and sequence of these scenarios, and addresses the issue of labelling and recognizing different intruder's activities. These intruder's activities, e.g. Intruder 1 behind the door, Intruder 1 opening the door, etc., are used for evaluation. It is measured, e.g., if these activities are detected by the framework at least once, to evaluate the detection quality of each feature.

5.1 Scenarios

For the scenarios an office was furnished with sensors in three different settings. Figure 5.1 shows the exact setup of Scenario 1, including the distribution of the sensors and the position of the stakeholders. In the middle of the room stood a desk the size of 150 cm x 75 cm. At this table sat User 1, equipped with a working laptop, which used the built-in webcam and microphone and the bluetooth sensor (for the exactly used hardware, see Appendix 7.2). At the same time this laptop was that on which the framework was running during the scenario. Additionally, for the used Smartphone Extension feature, an android smartphone was placed on the desk. Both cameras of the used devices pointed to the door behind User 1. The distance between the door and the table was exactly 345.50 cm. Intruder 1 was equipped with a second smartphone, which had bluetooth enabled. Intruder 1 entered a total of four times the room, whereby she was coming closer to the laptop each time (at the first time she stood at mark 1, at the second time at mark 2 and so on).

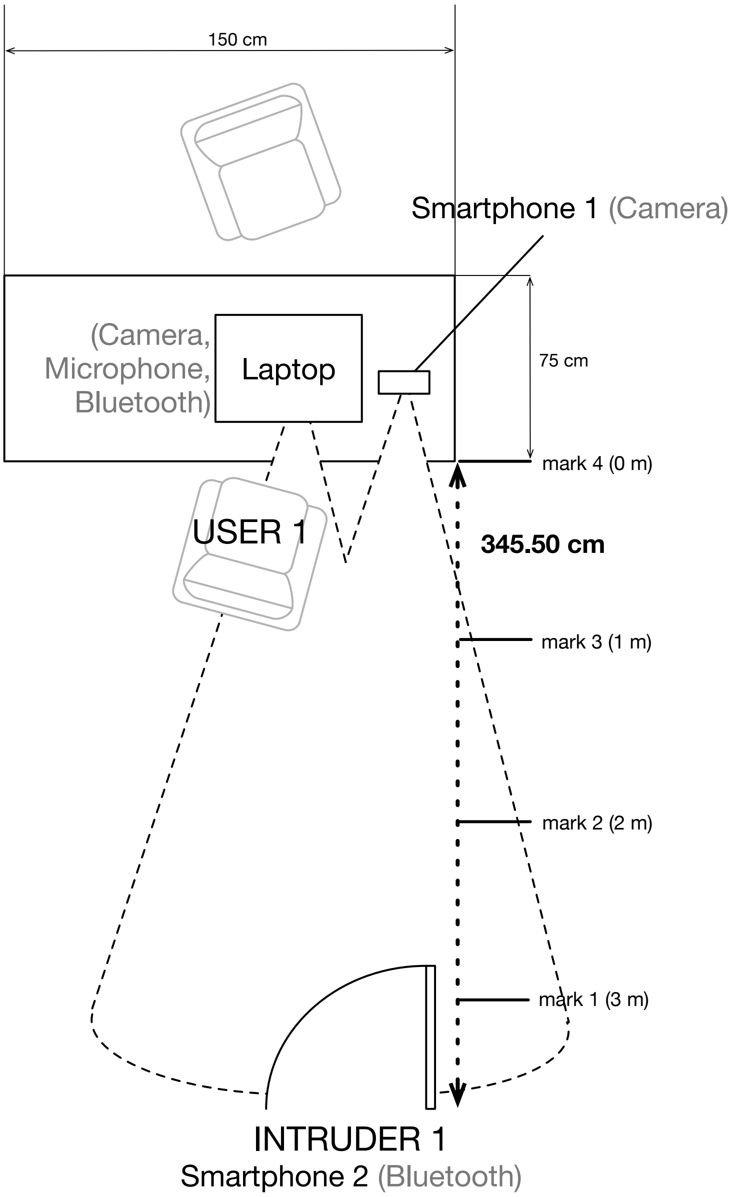


Figure 5.1: Sensors and Stakeholders in Scenario 1

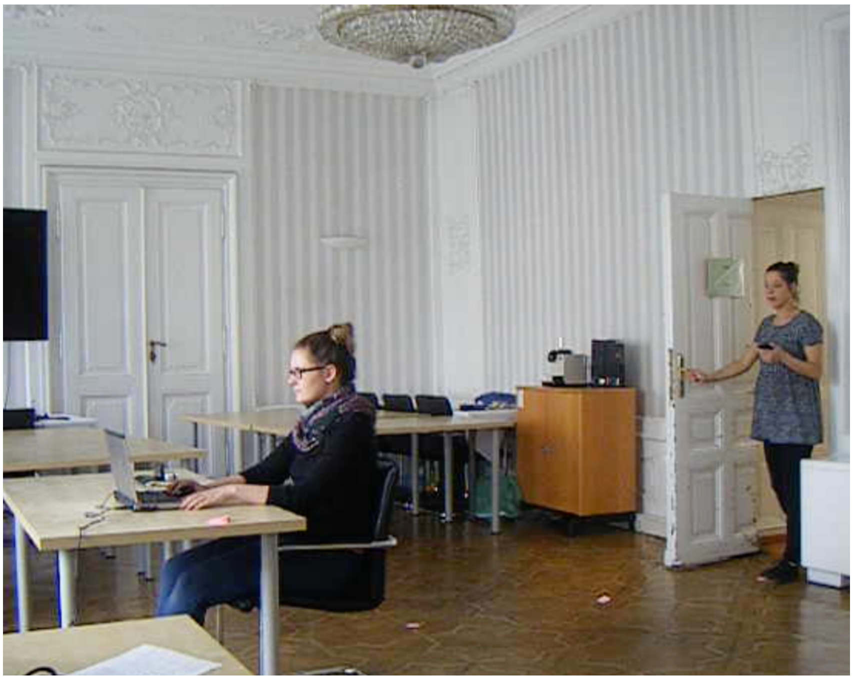


Figure 5.2: Photo from Evaluation Scenario 1

Figure 5.2 shows a photo taken from Scenario 1, where User 1 sits at the desk and Intruder 1 is coming through the door into the room, holding Smartphone 2 with activated bluetooth sensor. On the floor, the marks 1 - 4 are visible. The picture shows that from this angle almost the entire room including work station and two doors can be observed by one camera.

The setup of Scenario 2 (Figure 5.3) was very similar to the setup of Scenario 1. One big difference was that User 1 is no longer sitting with her back to the door but on the other side of the table. This means that the door was no longer in sight of the webcam of the laptop and only the camera of the smartphone has been directed to the door. So an entrance of Intruder 1 could only be detected visually by the motion detection of the Smartphone Extension feature.

While in Scenario 1 next to User 1, the door behind her has been under surveillance by the face recognition, in Scenario 2 only the user is in the webcam field of view. Thus it can not be used to detect an intruder directly in this way, it could be used for either a whitelisting approach (“four-eyes principle”, see explanation in Section 4.3) or for a continuous check if the user is still sitting at the work station.

The setup of Scenario 2 was deliberately chosen so that it is for example very similar to the situation in a bank, when a bank consultant receives customers in his office. Here it can be detected by very trivial methods like motion or sound detection, whether a customer enters the office or talks to the consultant.

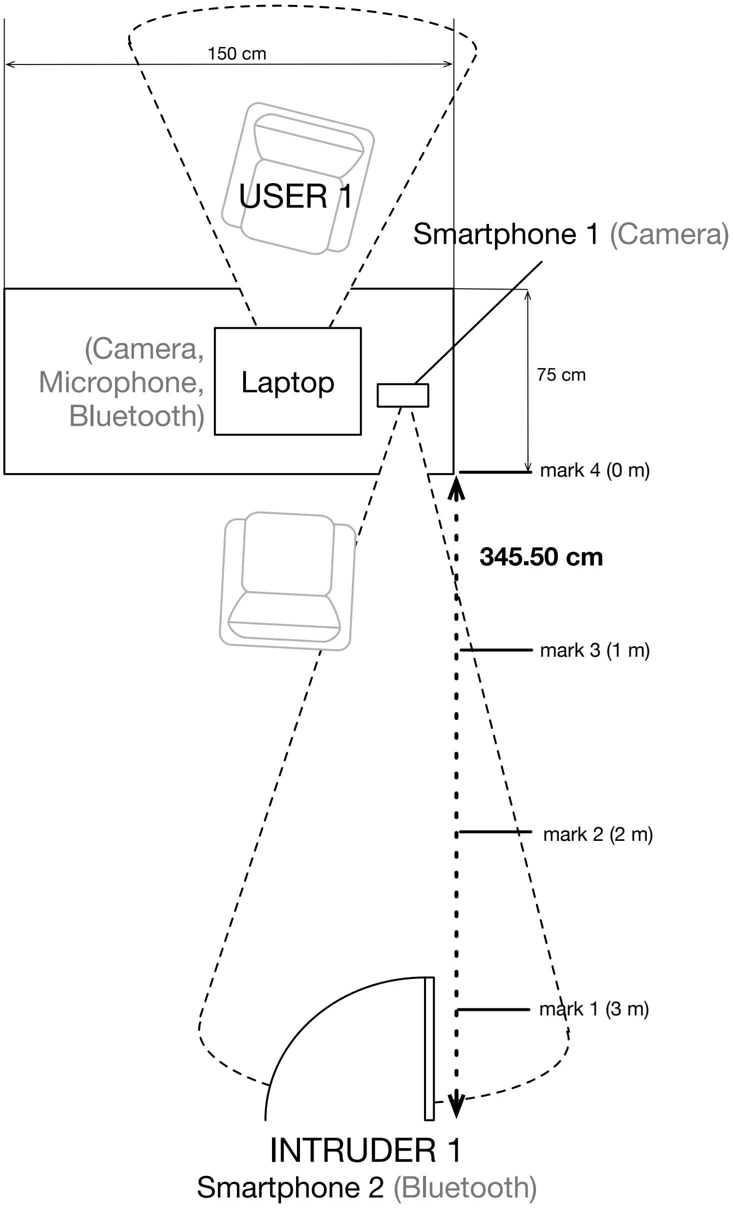


Figure 5.3: Sensors and Stakeholders in Scenario 2

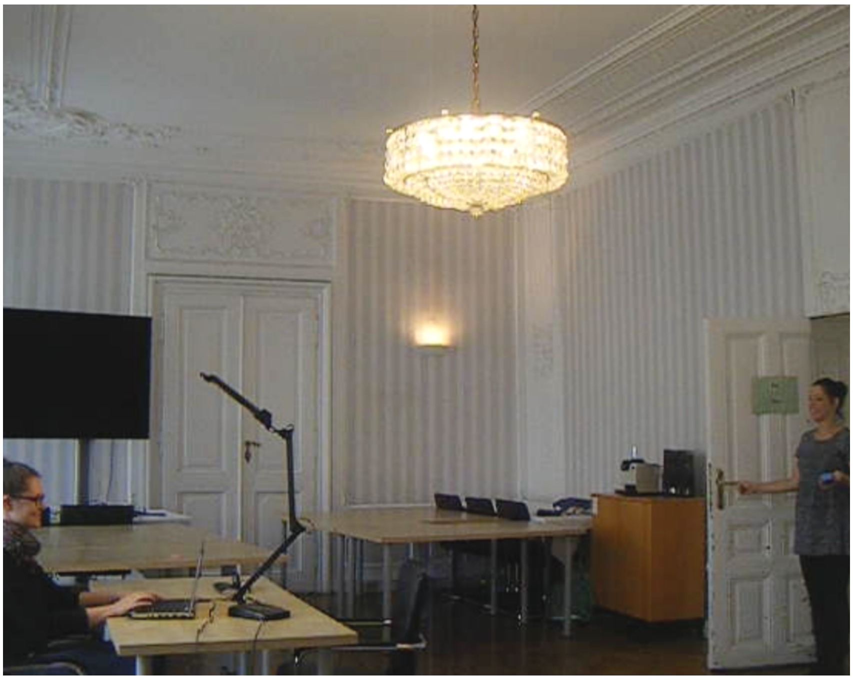


Figure 5.4: Photo from Evaluation Scenario 2

Figure 5.4 shows again a photo, now taken from Scenario 2. User 1 again is sitting at the desk with the laptop, Intruder 1 is entering the room. Due to lighting conditions, the ceiling lamp was turned on and an additional desk lamp was placed on the desk.

Scenario 3 was the most sophisticated scenario. An additional external webcam was used for the laptop, to cover a larger area in the room and being able to observe a second door. So User 1 is sitting as in Scenario 2, Smartphone 1 is still pointing to the door, where Intruder 1 is entering the room multiple times. In Scenario 3 a second Intruder is entering the room, as seen in Figure 5.5. He enters the room time-displaced with Intruder 1 and also stops at four different marks. Further in this scenario Intruder 2 is holding the Smartphone with enabled bluetooth sensor, instead of Intruder 1. With Scenario 3 it was intended to check whether two intruders can be detected simultaneously by the framework. Figure 5.6 again shows a photo taken from Scenario 3. Due to the complex setting, it is not longer possible to cover the whole scenario with only one camera.

According to the used features for the framework *Confidential Desktop*, wich were very simple, also the three evaluation scenarios were kept simple to show the feasibility of the proposed approach. The most sophisticated scenario observes two doors and thus two intruders entering the room simultaneously. Furthermore, the construction of the scenarios should be very close to office structures in practice.

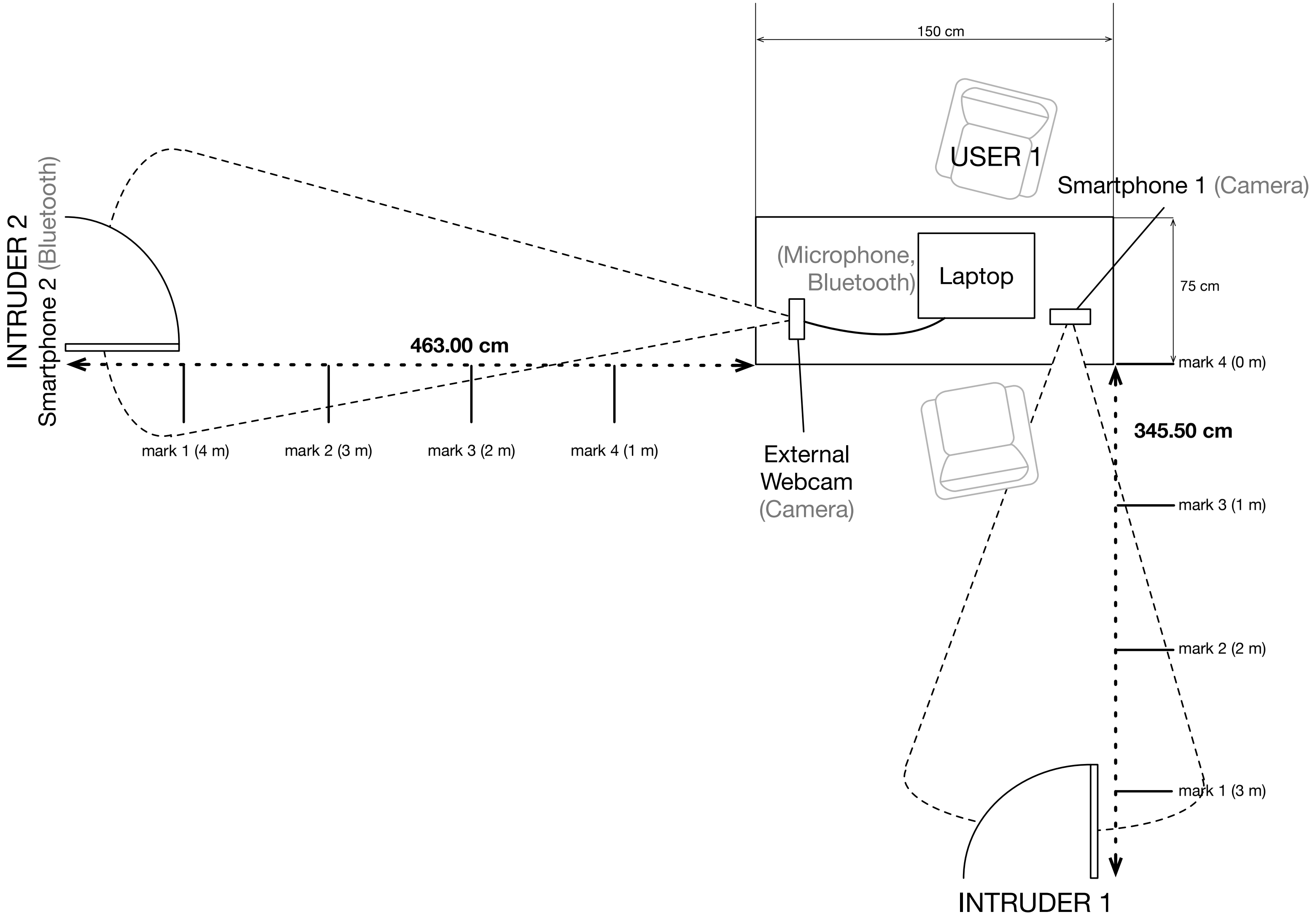


Figure 5.5: Sensors and Stakeholders in Scenario 3

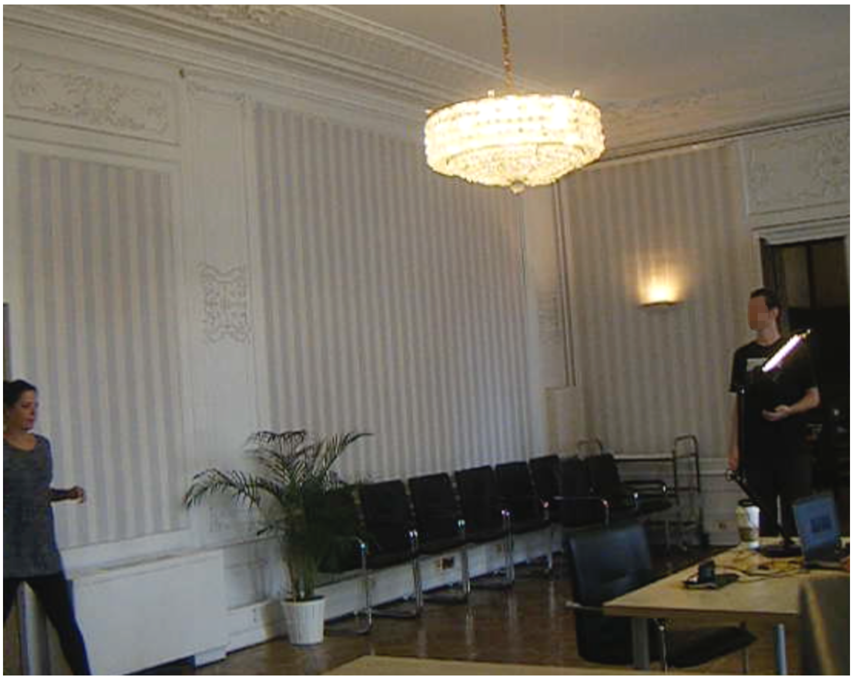


Figure 5.6: Photo from Evaluation Scenario 3

Having defined the structure of the three scenarios, now the execution is described. Figure 5.7 shows an Unified Modeling Language (UML) statechart [G^{+}09, Har87], which illustrates the sequence of the first two scenarios. Every scenario starts with the intruder behind the door. Then the intruder knocks on the door and the user responds with *“Ja”* (“Yes” in german). After that, the intruder opens the door and enters the room. If intruder enters the room the first time, she stops at mark 1 and says *“Mark 1, Mark 1”*. Then, after some seconds, she leaves the room, shuts the door and is outside of the room again. This procedure is repeated for every mark. After mark 4, the scenario is finished.

Basically, Scenario 3 is executed the same way as Scenario 1 and 2. The only difference in Scenario 3 is, that a second intruder is involved and he performs the exact same activities like the first intruder. So Intruder 1 knocks at the door, the user says *“Ja”*. Intruder 1 enters the room and speaks. Then, during Intruder 1 is standing at the current mark, Intruder 2 knocks at the second door. User says again *“Ja”*, and Intruder 2 enters the room and is speaking. Then after some seconds, both intruders are leaving the room simultaneously. This is repeated for every mark in the room. While Intruder 1 is detected by an external webcam with face and motion detection, Intruder 2 is only detected by the smartphone with motion detection.

In the next Section, the Intruder’s Activities for this three scenarios are described in detail.

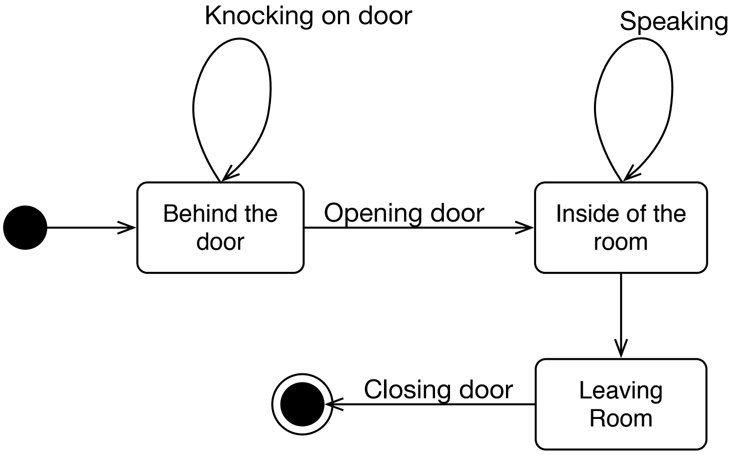


Figure 5.7: UML Statechart: Execution of Scenario 1 and 2

5.2 Labelling Intruder's Activities

The previous section has described the exact structure and execution of the scenarios used for the evaluation. In general, for the proper execution of the framework, a prior knowledge about the layout of the room, resp. the recognizable intruder's activities have to be defined.

The following list shows all intruder's activities used for the evaluation:

- Intruder 1 behind the door
- Intruder 1 knocking on the door
- Intruder 1 opening the door
- Intruder 1 inside of the room
- Intruder 1 speaking
- Intruder 1 leaving the room
- Screen in the field of vision of Intruder 1 (only Scenario 2)
- Intruder 2 behind the door (only Scenario 3)
- Intruder 2 knocking on the door (only Scenario 3)
- Intruder 2 opening the door (only Scenario 3)
- Intruder 2 inside of the room (only Scenario 3)

- Intruder 2 speaking (only Scenario 3)
- Intruder 2 leaving the room (only Scenario 3)

At this point it must be mentioned that, once the face of the intruder appears in the door, the activities "Opening the door" and "Inside of the room" can not be distinguished clearly from the framework and could therefore be grouped into the activity "Entering the room".

As already discussed in Chapter 4, the intruder's activities are predicted based on the implemented features. So different sets of features can detect a certain activity (Figure 5.8). For example, the activity "Behind the door" can only be detected by the Bluetooth Device Discovery feature, the activity "Speaking" only by the Noise Detection feature and the activity "Inside the room" can be detected by face detection in combination with motion detection and bluetooth device discovery.

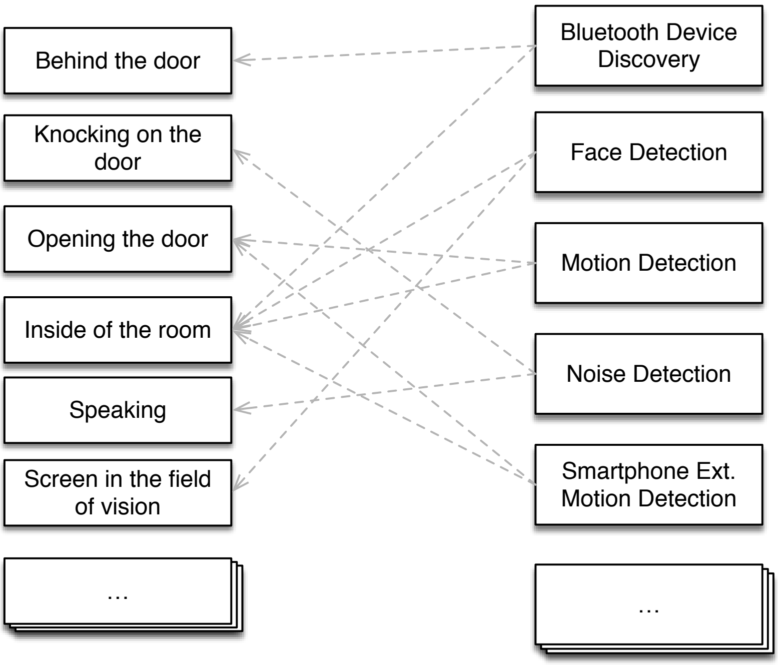


Figure 5.8: Mapping Features to Intruder's Activities

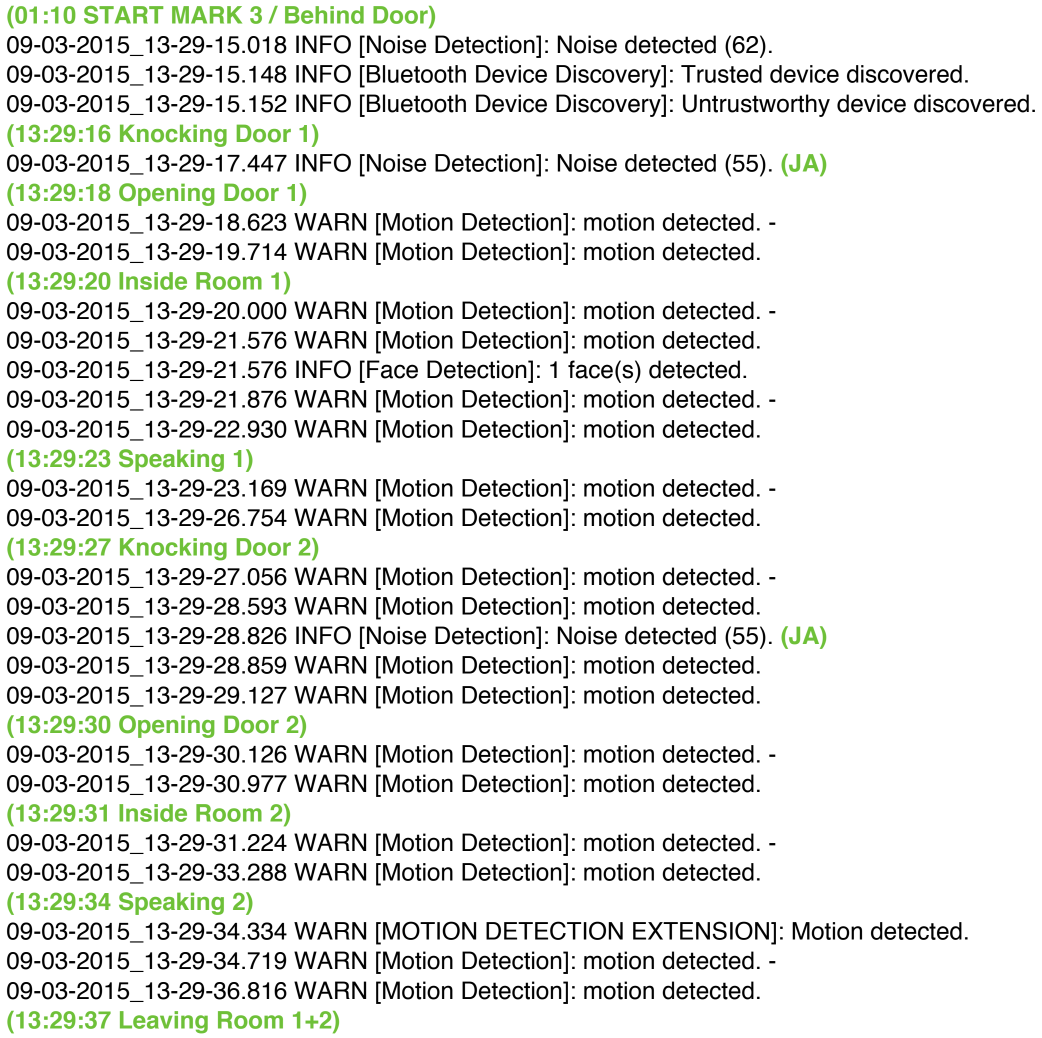
All these combinations of features, mapped to certain recognizable activities depend on the complexity of the integrated features and on the office layout. Because the framework is designed to be dynamically and easily expandable, it can be adapted to any office scenario.

Of great importance for the evaluation are the in Chapter 4 mentioned output files, which were created by the framework during the scenarios. Every time a feature has fired a detected activity, this was written by the framework in an output file including the accordingly timestamp. For higher accuracy and traceability the timestamps contain next to the date the time including seconds and milliseconds. The recorded logs were kept simple for each feature. For example, if the face detection detected something, it was recorded how many faces were detected by the camera. Here it would be possible to log information about the position of the face in the scene, or, by using an advanced algorithm it would be possible to give information if the detected face was female or male. In the case of detected bluetooth devices, the bluetooth device discovery generated one log for every found bluetooth device and gave additional information about whether the device was trusted, i.e., at least once paired with the laptop in the past, or untrusted. With the already used implementation it would be further possible to log information about the devices, like name and bluetooth address. Listing 7.10 shows an excerpt from the output file of Scenario 1.

Listing 5.1: Excerpt output file from Scenario 1

```
09-03-2015_11-48-51.341 INFO [Noise Detection]: Noise detected (56).
09-03-2015_11-48-57.241 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_11-48-57.266 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_11-48-57.274 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_11-49-00.878 INFO [Noise Detection]: Noise detected (57).
09-03-2015_11-49-03.972 WARN [Face Detection]: 2 face(s) detected.
09-03-2015_11-49-15.948 INFO [Face Detection]: 1 face(s) detected.
09-03-2015_11-49-24.800 INFO [Noise Detection]: Noise detected (60).
09-03-2015_11-49-28.287 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_11-49-28.291 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_11-49-28.298 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_11-49-37.594 WARN [Face Detection]: 2 face(s) detected.
09-03-2015_11-49-50.953 INFO [Face Detection]: 1 face(s) detected.
09-03-2015_11-49-53.027 WARN [Face Detection]: 2 face(s) detected.
09-03-2015_11-49-55.110 INFO [Face Detection]: 1 face(s) detected.
09-03-2015_11-49-59.438 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_11-49-59.443 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_11-49-59.451 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_11-50-03.137 INFO [Noise Detection]: Noise detected (59).
09-03-2015_11-50-03.264 INFO [Noise Detection]: Noise detected (61).
09-03-2015_11-50-11.846 INFO [Noise Detection]: Noise detected (56).
09-03-2015_11-50-11.974 INFO [Noise Detection]: Noise detected (55). (JA)
09-03-2015_11-50-15.590 WARN [Face Detection]: 2 face(s) detected.
09-03-2015_11-50-18.659 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_11-50-28.633 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_11-50-28.045 INFO [Face Detection]: 1 face(s) detected.
09-03-2015_11-50-28.953 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_11-50-30.167 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_11-50-30.542 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_11-50-30.546 INFO [Bluetooth DD]: Untrustworthy device discovered.
```

Additionally to the generated output “log” files, a video was taken from every scenario and frames of the webcam video, which was used as input for face and motion detection, were stored. These video frames include detection bounding boxes in the case of found faces or motion, as visualized in Figure 4.7. Thus it can be traced in retrospect any time, which was recognized by the feature as face or movement. For evaluation, the detected activities from the features were compared with the actually performed intruder’s activities to evaluate the performance of the detection.



```
(01:10 START MARK 3 / Behind Door)
09-03-2015_13-29-15.018 INFO [Noise Detection]: Noise detected (62).
09-03-2015_13-29-15.148 INFO [Bluetooth Device Discovery]: Trusted device discovered.
09-03-2015_13-29-15.152 INFO [Bluetooth Device Discovery]: Untrustworthy device discovered.
(13:29:16 Knocking Door 1)
09-03-2015_13-29-17.447 INFO [Noise Detection]: Noise detected (55). (JA)
(13:29:18 Opening Door 1)
09-03-2015_13-29-18.623 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-19.714 WARN [Motion Detection]: motion detected.
(13:29:20 Inside Room 1)
09-03-2015_13-29-20.000 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-21.576 WARN [Motion Detection]: motion detected.
09-03-2015_13-29-21.576 INFO [Face Detection]: 1 face(s) detected.
09-03-2015_13-29-21.876 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-22.930 WARN [Motion Detection]: motion detected.
(13:29:23 Speaking 1)
09-03-2015_13-29-23.169 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-26.754 WARN [Motion Detection]: motion detected.
(13:29:27 Knocking Door 2)
09-03-2015_13-29-27.056 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-28.593 WARN [Motion Detection]: motion detected.
09-03-2015_13-29-28.826 INFO [Noise Detection]: Noise detected (55). (JA)
09-03-2015_13-29-28.859 WARN [Motion Detection]: motion detected.
09-03-2015_13-29-29.127 WARN [Motion Detection]: motion detected.
(13:29:30 Opening Door 2)
09-03-2015_13-29-30.126 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-30.977 WARN [Motion Detection]: motion detected.
(13:29:31 Inside Room 2)
09-03-2015_13-29-31.224 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-33.288 WARN [Motion Detection]: motion detected.
(13:29:34 Speaking 2)
09-03-2015_13-29-34.334 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_13-29-34.719 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-36.816 WARN [Motion Detection]: motion detected.
(13:29:37 Leaving Room 1+2)
```

Figure 5.9: Excerpt evaluation sheet from Scenario 3

Figure 5.9 shows an excerpt of the annotated evaluation sheet from Scenario 3. The evaluation sheets include the generated output files and further information about the performed intruder's activities, which were extracted on the basis of the recorded videos of the scenarios. During the performance of each repetition of every scenario, it was important to be able to synchronize the framework session and the according video recording in the later evaluation. Therefore as initiation of every iteration, a "knocking sync signal" on the table was used, which, on the one hand, can be seen in the video recording and, on the other hand, which was discovered by the framework with the help of the integrated noise detection feature. The performed intruder's activities including their timestamp were added to the output files in color so that they visually stand out from the framework logs. With the help of the evaluation sheets it can be identified at a glance, which activities were discovered by the framework and thus, which features worked well. The complete evaluation sheets for all scenarios can be found in Appendix 7.3.

5.3 Activity Recognition Performance

Having discussed the scenarios and the intruder's activities, this section shows the final performance of the algorithm recognition performance, resp. the quality of the detection of the intruder's activities. The following evaluation methods are based in simplified form on the evaluation of the thesis of Emmanuel Munguia Tapia [MT03]. He also described in his thesis an activity recognition system using ubiquitous sensors and discusses in his evaluation the quality of the detection of various activities in a private household. Therefore, for the evaluation of the proposed framework *Confidential Desktop*, two different methods, which were presented by Tapia among others, are used to measure the performance of the activity detection [MT03]:

- **Activity detected in best interval** Measures whether the activity is detected in a predefined interval. Since intruder's activities in an access control framework should be recognized the best as soon as possible, the interval is the first 0.5 seconds after the start of the activity.
- **Activity detected at least once** Measures whether the activity is detected at least once while the activity is actually taking place. Further no delay is allowed.

Since the general approach of Tapia's work is similar to the proposed framework *Confidential Desktop*, these performance indicators are well suited to illustrate the results of all three scenarios. The data in the tables in the following Section are calculated from the average detection rate of the intruder's activities in all repetitions of a scenario. If it was detected in all repetitions that the intruder is located behind the door, this results in a value of 1.00, if it was never detected when the intruder was speaking, the result is 0.00. In the case of the activities "Inside Room" and "Leaving Room" in Scenario 1, where the two features for face and motion detection should fire, it has been considered that both features fire or just one. It was determined that the facial recognition in any case has a higher weighting than the motion detection. If only the face detection fires and motion detection does not, this gives a detection rate of 0.75.

5.4 Results

The following tables show the evaluation results for all three scenarios, if the recognizable intruder's activities were detected at least once by the framework (Table 5.1), and if the activities were detected in the best interval (Table 5.2).

Intruder's Activities	Scenario 1	Scenario 2	Scenario 3
Intruder 1 Behind the door	**1.00**	**1.00**	-
Intruder 1 Knocking on door	0.75	**1.00**	0.50
Intruder 1 Opening door	0.00	0.00	0.00
Intruder 1 Inside room	**0.86**	0.60	0.50
Intruder 1 Speaking	0.00	0.00	0.00
Intruder 1 Leaving room	**0.86**	0.60	0.50
Screen in field of vision of Intruder 1	-	**1.00**	-
Intruder 2 Behind the door	-	-	**1.00**
Intruder 2 Knocking on door	-	-	0.75
Intruder 2 Opening door	-	-	**1.00**
Intruder 2 Inside room	-	-	**1.00**
Intruder 2 Speaking	-	-	0.00
Intruder 2 Leaving room	-	-	**1.00**

Table 5.1: Activity detected at least once

Intruder’s Activities	Scenario 1	Scenario 2	Scenario 3
Intruder 1 Behind the door	**1.00**	**1.00**	-
Intruder 1 Knocking on door	**0.56**	**0.75**	0.38
Intruder 1 Opening door	0.00	0.00	0.00
Intruder 1 Inside room	0.19	0.00	0.00
Intruder 1 Speaking	0.00	0.00	0.00
Intruder 1 Leaving room	0.00	0.00	0.00
Screen in field of vision of Intruder 1	-	0.00	-
Intruder 2 Behind the door	-	-	**1.00**
Intruder 2 Knocking on door	-	-	**0.56**
Intruder 2 Opening door	-	-	0.25
Intruder 2 Inside room	-	-	0.25
Intruder 2 Speaking	-	-	0.00
Intruder 2 Leaving room	-	-	0.25

Table 5.2: Activity detected in best interval

5.5 Discussion

The results, which were presented in the previous Section, show that the quality of features varies greatly. By far the most reliable feature is the bluetooth device discovery feature, which has continuously recognized that an untrusted device is near. But this must be treated with caution, however, since the range of the bluetooth sensor can be up to 100 m high, this can lead to false positives. Noise detection has worked worst. Generally, it has only detected when the user replied to the knock on the door with *“JA”*. The knocking on the door and speaking of both intruders has not been identified in any instance. This could be due to the range of the built-in microphone as well as the algorithm of the sound recognition.

Faces are generally well recognized by the framework, even at long distance i.e., intruders are detected by the face detection feature immediately when their faces show up in the doorway. However faces must be completely visible in the frame, so that they are recognized by the face recognition. So it can happen that no face is detected even in the case of the intruder's activity "Screen in the field of view of the intruder", since only a part of the face is covered by the webcam. One possible solution here would be to implement an additional face tracking algorithm to support the face detection feature.

The motion detection feature also runs flawlessly and detects without problems, if the door is opened as well as the intruder entering and staying in the room as soon as he or she leaves the room again. However, the smartphone extension feature with another motion detection algorithm implemented only detected motion from about mark 3 (1 m distance to the desk) in all instances i.e., the opening of the door was not recognized and the intruder only on closer approach to the workplace. This could be due to the implemented method and can be improved by adjusting parameters.

The results of the evaluation, whether the activities are discovered in the best interval are significantly worse than the results of the evaluation, whether the activities are detected at least once. The 0.5 seconds for the best interval were deliberately very strictly selected in order to check how quickly the features detect an intruder and especially how quickly the framework reacts to it. The majority of the activities taking place are recognized by the features and processed by the framework within 0.6 seconds. Furthermore, the speed of the recognition is lower, when two different features are needed to recognize something in order to guarantee the occurrence of an intruder's activity. Here has been found that, for example, the motion detection feature reacts much faster than the face detection feature.

Generally speaking, the quality of the functioning of the framework is highly dependent on the implemented features. This thesis has shown that it is possible even with non-sophisticated algorithms and standard office hardware to identify intruders in real-time in order to control the access to system resources.

CHAPTER 6

Conclusion and Perspectives

6.1 Contribution

This thesis described a new approach for a context-aware access control framework based on environmental conditions in an ubiquitous office environment. The framework, called *Confidential Desktop* continuously checks whether the user is alone in the office or not. This is done by detecting predefined intruder's activities like "Knocking on the door", "Inside the room" or "Speaking" by the use of different integrated features for face and motion detection, bluetooth device discovery and sound detection. By integrating various interacting features, intruder's activities can be detected in real-time and the access to security-critical system resources can be limited.

After background information about state-of-the-art methods in areas like computer vision and audio analysis were gathered, state-of-the-art algorithms were implemented as features for the framework. Further a smartphone application for Android was implemented to equip the framework with an additional, external motion detection feature. The framework *Confidential Desktop* itself was implemented as a desktop Java application, including a graphical interface, in order to show feasibility. For the evaluation, three different scenarios in a simulated office environment were performed and the detection quality of each integrated feature was evaluated, as well as the interaction of all components of the framework.

This thesis has shown that even with non-sophisticated features and standard office hardware, the intrusion of an attacker can be detected by the framework. When an intruder is detected the framework limits the access to security-critical or sensitive data, thus displayed information on computer screens can be protected from prying eyes and confidentiality can be preserved.

6.2 Future Work

Evaluation has shown that the quality of the functionality of the framework is dependent on the detection quality of its features. Thus the framework was designed to be highly expandable, new and more sophisticated features can be integrated easily. Possible features could be e.g. more complex methods for human detection and tracking. More external devices can be used to cover the whole office and different scenarios. Also additional sensors can help to improve the detection quality of intruder's activities, e.g. motion sensors and microphones directly located at doors.

In addition information about the detection of the identity of a person can improve the quality of the framework. If for example one feature detects that the user is not sitting at his workstation any more, there is a high probability that the person opening or closing the door is the user itself. Generally not only identifying that somebody is coming into the office but also to authenticate him or her in order to gain additional information for the framework.

CHAPTER 7

Appendices

7.1 Implementation

The following listings show relevant code excerpts from the implemented *Confidential Desktop* framework and are explained in Chapter 4 (System Design and Implementation).

Listing 7.1: Initiating feature and loading last status from preferences

```java
private static MainWindow mainWindow = MainWindow.getInstance();
private static FeaturePreferences featurePreferences =
                    FeaturePreferences.getInstance();

(...)
if (featurePreferences.isNoiseDetectionEnabled())
                    setNoiseDetectionFeatureActive(true);

(...)
public static void setNoiseDetectionFeatureActive(boolean active) {
    NoiseDetectionFeature noiseDetectionFeature =
                    NoiseDetectionFeature.getInstance();

    if (active) {
        mainWindow
            .printConsoleText("Feature␣NOISE␣DETECTION␣enabled."
                    , false);
        noiseDetectionFeature.start();
    } else {
        noiseDetectionFeature.stop();
        mainWindow
            .printConsoleText("Feature␣NOISE␣DETECTION␣disabled."
                    , false);
    }
}
```

Listing 7.2: Writing all events (incl. timestamps) from a session into a file

```
private FileWriter() {
    (...)
    File file = new File(currentSession
        + "_ConfidentialDesktop.txt");
    try {
        output = new BufferedWriter(new FileWriter(file));
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private String getTimestamp() {
    Date currentTimestamp = new Timestamp(Calendar.getInstance()
        .getTime().getTime());
    return new SimpleDateFormat("MM-dd-yyyy_HH-mm-ss.SSS")
        .format(currentTimestamp);
}

(...)
public void print(String message) {
    try {
        (...)
        output.write(getTimestamp() + "␣" + message + "\n");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Listing 7.3: Saving all video frames from a session into a folder

```
public void newImageFolder() {
    String newSessionTimestamp = getTimestamp();
    boolean b = new File(PATH + newSessionTimestamp).mkdirs();

    if(b) {
        currentSession = newSessionTimestamp;
    }
}

public void saveImage(BufferedImage buff, String type) {
    try {
        ImageIO.write(buff, "jpg", new File(PATH
                + currentSession + "\\"
                + type + "_" + currentTimestamp
                + ".jpg"));
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Listing 7.4: Managing application processes

```
private Process browser;
private ProcessExitDetector browserExitDetector;

(...)
printConsoleText("Internet␣Browser␣requested␣...", false);

(...)
printConsoleText("Internet␣Browser␣opening␣...", false);

try {
    browser = new ProcessBuilder(WIN8_BROWSER_PATH).start();
    browserExitDetector = new ProcessExitDetector(browser);
    browserExitDetector .addProcessListener(process -> {
        if(browserRunning) {
            printConsoleText("Internet␣Browser␣was␣closed."
                , false);
            browserRunning = false;
            runningProcesses--;
            printNumberOfProcesses();
        }
    });

    browserExitDetector.start();
    browserRunning = true;
    runningProcesses++;
    printNumberOfProcesses();
} catch (IOException e) {
    e.printStackTrace();
    printConsoleText("IO␣ERROR:␣Could␣not␣start␣Internet␣Browser."
            , true);
}
```

Listing 7.5: Quartz Scheduler for Bluetooth Device Discovery

```
BluetoothDeviceDiscovery bluetoothDeviceDiscovery =
    new BluetoothDeviceDiscovery();
Scheduler scheduler = new StdSchedulerFactory().getScheduler();

JobDetail discoveryJob =
    newJob(bluetoothDeviceDiscovery.getClass())
        .withIdentity("discoveryJob", "bluetooth")
        .build();

Trigger discoveryJobTrigger = newTrigger()
    .withIdentity("discoveryJobTrigger", "bluetooth")
```

```
        .startNow()
        .withSchedule(simpleSchedule()
            .withMisfireHandlingInstructionIgnoreMisfires()
            .withIntervalInSeconds(20)
            .repeatForever())
        .build();

if(!scheduler.checkExists(discoveryJob.getKey()))
    scheduler.scheduleJob(discoveryJob, discoveryJobTrigger);
```

Listing 7.6: Bluetooth Device Discovery

```
/**
 * adapted from ...
 * http://www.jsr82.com/jsr-82-sample-device-discovery/
 */

 @Override
public void execute(JobExecutionContext jobExecutionContext)
    throws JobExecutionException {
    LocalDevice localDevice = null;

    if(vecDevices != null && !vecDevices.isEmpty()) {
        vecDevices.clear();
    }
    try {
        localDevice = LocalDevice.getLocalDevice();
    } catch (BluetoothStateException e) {
        e.printStackTrace();
    }
    assert localDevice != null;

    //find devices
    DiscoveryAgent agent = localDevice.getDiscoveryAgent();

    try {
        agent.startInquiry(DiscoveryAgent.GIAC, this);
    } catch (BluetoothStateException e) {
        e.printStackTrace();
    }
    try {
        synchronized (lock) {
            lock.wait();
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    //print all devices in vecDevices
    int deviceCount = vecDevices.size();
    if (deviceCount <= 0) {
        mainwindow.printConsoleText(
        "INFO␣[Bluetooth␣DD]:␣No␣Devices␣found."
        , false);
    }
    agent.cancelInquiry(this);
}

@Override
public void deviceDiscovered(RemoteDevice remoteDevice
    , DeviceClass deviceClass) {

    if(remoteDevice.isTrustedDevice()) {
        mainwindow.printConsoleText(
        "INFO␣[Bluetooth␣DD]:␣" +
            "Trusted␣device␣discovered."
            , false);
    }
    else {
        mainwindow.printConsoleText(
        "WARN␣[Bluetooth␣Device␣Discovery]:␣" +
            "Untrustworthy␣device␣discovered."
            , true);
    }
    //add the device to the vector
    if (!vecDevices.contains(remoteDevice))
        vecDevices.addElement(remoteDevice);
}
```

Listing 7.7: Face Detection

```
/**
 * adapted from ...
 * https://github.com/emara-geek/
 * real-time-face-detection-using-opencv-with-java
 */

System.loadLibrary("opencv_java300");

VideoCapture webSource = new VideoCapture(0);
//0 = internal webcam; 1 = external webcam
Mat frame = new Mat();
MatOfByte mem = new MatOfByte();
CascadeClassifier faceDetector = new CascadeClassifier(
    FaceDetectionFeature.class.getResource(
    "haarcascade_frontalface_alt.xml").getPath().substring(1));
MatOfRect faceDetections = new MatOfRect();

if (webSource.grab()) {
    webSource.retrieve(frame);
    faceDetector.detectMultiScale(frame, faceDetections);
    checkFaces(faceDetections.toArray().length);
    Graphics g = faceDetectorDialog
        .getFaceDetectorDialogPanel()
        .getGraphics();

    for (Rect rect : faceDetections.toArray()) {
        Imgproc.rectangle(frame, new Point(rect.x, rect.y),
        new Point(rect.x + rect.width, rect.y + rect.height),
        new Scalar(255, 0, 0));
    }
    Imgcodecs.imencode(".bmp", frame, mem);
    Image im = ImageIO
                .read(new ByteArrayInputStream(mem.toArray()));
    BufferedImage buff = (BufferedImage) im;
    g.drawImage(buff, 0, 0, faceDetectorDialog.getWidth(),
    faceDetectorDialog.getHeight() - 150, 0, 0, buff.getWidth(),
    buff.getHeight(), null)
}
```

7.2 Setup

Implementation and all results of the evaluation are based on the following hard- and software. Further all technologies and libraries used from the features and also the exemplarily used applications are listed.

7.2.1 Hardware (Notebook DELL Inc. Vostro 3350)

- **CPU** 2.30 GHz Intel Core i5-2410M
- **Memory** 8 GB
- **Graphics** Intel HD Graphics 3000
- **Storage** 297 GB HDD
- **Bluetooth** BT V3.0+HS
- **Built-in Microphone** Dell Array Microphones
- **Built-in Webcam** Dell HD Webcam (2 MP)
- **External Webcam** Logitech HD Webcam C270 (3 MP)

7.2.2 Hardware (Smartphone Nexus S)

- **CPU** ARMv7 Processor rev 2 (v7l)
- **Memory** 383 MB

7.2.3 Software

- Windows 8.1 Enterprise (64-bit)
- Java 1.8.0_45
- Swing (GUI Widget Toolkit)
- Bluecove-2.1.1 (Bluetooth Discovery)
- Quartz-2.2.1 (Bluetooth Discovery)
- OpenCV-300 for Java (Face & Motion Detection)
- Java Speech API (Noise Detection)
- Android 4.3.1
- Android API 22 Platform

7.2.4 Applications

- Mozilla Firefox ESR 38.2.1
- Microsoft Word 2013 (15.0.4745.1001)
- Microsoft Outlook 2013 (15.0.4745.1000)
- GIMP 2.8.14

7.3 Evaluation Sheets

The complete evaluation sheets used for evaluation. These include both the output files, generated by the framework and the actually performed intruder's activities, which were included into the files with the according timestamp. When one of the motion detection features has fired more than 2 times without a break, only the first and the last event are listed and connected by a hyphen.

Listing 7.8: Evaluation Sheet Scenario 1

```
(00:00 START MARK 1 / Behind Door)
09-03-2015_11-48-51.341 INFO [Noise Detection]: Noise detected (56).
09-03-2015_11-48-57.241 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_11-48-57.266 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_11-48-57.274 INFO [Bluetooth DD]: Untrustworthy device discovered.
(11:48:58 Knocking Door)
09-03-2015_11-49-00.878 INFO [Noise Detection]: Noise detected (57). (JA)
(11:49:01 Opening Door)
09-03-2015_11-49-03.972 WARN [Face Detection]: 2 face(s) detected.
(11:49:04 Inside Room)
(11:49:05 - 11:49:10 Speaking)
(11:49:11 Leaving Room) + (11:49:12 Closing Door) = Leaving Room
09-03-2015_11-49-15.948 INFO [Face Detection]: 1 face(s) detected.
(11:49:17 Behind Door)

(00:34 START MARK 2 / Behind Door)
09-03-2015_11-49-24.800 INFO [Noise Detection]: Noise detected (60).
09-03-2015_11-49-28.287 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_11-49-28.291 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_11-49-28.298 INFO [Bluetooth DD]: Untrustworthy device discovered.
(11:49:29 Knocking Door) (JA)
(11:49:34 Opening Door)
(11:49:37 Inside Room)
09-03-2015_11-49-37.594 WARN [Face Detection]: 2 face(s) detected.
(11:49:40 - 11:49:45 Speaking)
(11:49:49 Leaving Room)
09-03-2015_11-49-50.953 INFO [Face Detection]: 1 face(s) detected.
```

```
(11:49:52 Closing Door)
09-03-2015_11-49-53.027 WARN [Face Detection]: 2 face(s) detected.
09-03-2015_11-49-55.110 INFO [Face Detection]: 1 face(s) detected.
(11:49:56 Behind Door)
09-03-2015_11-49-59.438 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_11-49-59.443 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_11-49-59.451 INFO [Bluetooth DD]: Untrustworthy device discovered.

(01:12 START MARK 3 / Behind Door)
09-03-2015_11-50-03.137 INFO [Noise Detection]: Noise detected (59).
09-03-2015_11-50-03.264 INFO [Noise Detection]: Noise detected (61).
(11:50:08 Knocking Door)
09-03-2015_11-50-11.846 INFO [Noise Detection]: Noise detected (56).
09-03-2015_11-50-11.974 INFO [Noise Detection]: Noise detected (55). (JA)
(11:50:13 Opening Door)
(11:50:15 Inside Room)
09-03-2015_11-50-15.590 WARN [Face Detection]: 2 face(s) detected.
09-03-2015_11-50-18.659 WARN [MOTION DETECTION EXTENSION]: Motion detected.
(11:50:19 - 11:50:22 Speaking)
(11:50:27 Leaving Room)
09-03-2015_11-50-28.633 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_11-50-28.045 INFO [Face Detection]: 1 face(s) detected.
09-03-2015_11-50-28.953 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_11-50-30.167 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_11-50-30.542 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_11-50-30.546 INFO [Bluetooth DD]: Untrustworthy device discovered.
(11:50:31 Closing Door)

(01:50 START MARK 4 / Behind Door)
09-03-2015_11-50-40.886 INFO [Noise Detection]: Noise detected (64).
09-03-2015_11-50-41.014 INFO [Noise Detection]: Noise detected (57).
(11:50:44 Knocking Door)
09-03-2015_11-50-46.726 INFO [Noise Detection]: Noise detected (58). (JA)
(11:50:48 Opening Door)
(11:50:50 Inside Room)
09-03-2015_11-50-50.581 WARN [Face Detection]: 2 face(s) detected.
09-03-2015_11-50-54.069 WARN [MOTION DETECTION EXTENSION]: Motion detected. -
09-03-2015_11-50-55.183 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_11-50-55.269 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_11-50-55.273 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_11-50-55.297 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_11-50-55.521 WARN [MOTION DETECTION EXTENSION]: Motion detected. -
09-03-2015_11-50-56.401 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_11-50-54.737 INFO [Face Detection]: 1 face(s) detected.
09-03-2015_11-50-56.925 WARN [MOTION DETECTION EXTENSION]: Motion detected. -
09-03-2015_11-50-57.927 WARN [MOTION DETECTION EXTENSION]: Motion detected.
(11:50:58 Speaking)
09-03-2015_11-50-58.365 WARN [MOTION DETECTION EXTENSION]: Motion detected. -
09-03-2015_11-50-59.913 WARN [MOTION DETECTION EXTENSION]: Motion detected.
(11:51:00 Speaking)
09-03-2015_11-51-00.519 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_11-51-00.991 WARN [MOTION DETECTION EXTENSION]: Motion detected.
(11:51:01 Screen in field of vision)
09-03-2015_11-51-01.441 WARN [MOTION DETECTION EXTENSION]: Motion detected. -
09-03-2015_11-51-03.941 WARN [MOTION DETECTION EXTENSION]: Motion detected.
(11:51:06 Speaking)
09-03-2015_11-51-06.670 WARN [MOTION DETECTION EXTENSION]: Motion detected. -
09-03-2015_11-51-26.485 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_11-51-26.535 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_11-51-26.539 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_11-51-26.547 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_11-51-26.818 WARN [MOTION DETECTION EXTENSION]: Motion detected. -
09-03-2015_11-51-27.565 WARN [MOTION DETECTION EXTENSION]: Motion detected.
(11:51:28 Speaking START*)
09-03-2015_11-51-30.286 WARN [MOTION DETECTION EXTENSION]: Motion detected. -
09-03-2015_11-51-36.719 WARN [MOTION DETECTION EXTENSION]: Motion detected.
(*11:51:37 Speaking END)
09-03-2015_11-51-37.085 WARN [MOTION DETECTION EXTENSION]: Motion detected. -
09-03-2015_11-51-45.538 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_11-51-45.854 WARN [Face Detection]: 2 face(s) detected.
09-03-2015_11-51-47.856 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_11-51-57.906 WARN [MOTION DETECTION EXTENSION]: Motion detected.
(11:51:58 Leaving Room)
```

```
09-03-2015_11-51-58.205 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_11-51-58.567 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_11-51-58.571 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_11-51-58.776 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_11-51-58.797 INFO [Face Detection]: 1 face(s) detected.
09-03-2015_11-51-59.938 WARN [MOTION DETECTION EXTENSION]: Motion detected. -
09-03-2015_11-52-01.695 WARN [MOTION DETECTION EXTENSION]: Motion detected.
(11:52:03 Closing Door)
09-03-2015_11-52-03.777 WARN [Face Detection]: 2 face(s) detected.
09-03-2015_11-52-05.767 INFO [Face Detection]: 1 face(s) detected.
09-03-2015_11-52-06.904 WARN [Face Detection]: 2 face(s) detected.
(11:52:07 Behind Door)
09-03-2015_11-52-07.946 INFO [Face Detection]: 1 face(s) detected.
```

Listing 7.9: Evaluation Sheet Scenario 2

```
(00:00 START MARK 1 / Behind Door)
09-03-2015_12-33-10.258 INFO [Noise Detection]: Noise detected (65).
(12:33:13 Knocking Door)
09-03-2015_12-33-15.165 INFO [Noise Detection]: Noise detected (55). (JA)
09-03-2015_12-33-15.294 INFO [Noise Detection]: Noise detected (58).
09-03-2015_12-33-15.426 INFO [Noise Detection]: Noise detected (55).
(12:33:16 Opening Door) + (12:33:19 Inside Room) = Entering Room
(12:33:20 - 12:33:25 Speaking)
(12:33:27 Leaving Room) + (12:33:28 Closing Door) = Leaving Room
(12:33:32 Behind Door)

(00:26 START MARK 2 / Behind Door)
09-03-2015_12-33-36.606 INFO [Noise Detection]: Noise detected (62).
09-03-2015_12-33-36.734 INFO [Noise Detection]: Noise detected (59).
09-03-2015_12-33-36.737 INFO [Noise Detection]: Noise detected (55).
(12:33:38 Knocking Door)
09-03-2015_12-33-38.276 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_12-33-38.282 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_12-33-38.291 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_12-33-40.874 INFO [Noise Detection]: Noise detected (56). (JA)
09-03-2015_12-33-41.006 INFO [Noise Detection]: Noise detected (56).
(12:33:41 Opening Door)+ (12:33:43 Inside Room) = Entering Room
(12:33:47 - 12:33:53 Speaking)
(12:33:54 Leaving Room) + (12:33:57 Closing Door) = Leaving Room
(12:34:00 Behind Door)

(00:56 START MARK 3 / Behind Door)
09-03-2015_12-34-05.756 INFO [Noise Detection]: Noise detected (55).
09-03-2015_12-34-06.009 INFO [Noise Detection]: Noise detected (56).
09-03-2015_12-34-06.138 INFO [Noise Detection]: Noise detected (58).
09-03-2015_12-34-06.141 INFO [Noise Detection]: Noise detected (55).
(12:34:08 Knocking Door)
09-03-2015_12-34-09.312 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_12-34-09.316 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_12-34-10.787 INFO [Noise Detection]: Noise detected (57). (JA)
09-03-2015_12-34-10.915 INFO [Noise Detection]: Noise detected (55).
(12:34:11 Opening Door) + (12:34:13 Inside Room) = Entering Room
(12:34:16 Speaking START*)
09-03-2015_12-34-16.092 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_12-34-16.606 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_12-34-33.845 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_12-34-33.849 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_12-34-44.021 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_12-34-44.653 WARN [MOTION DETECTION EXTENSION]: Motion detected.
(*12:34:51 Speaking END)
(12:34:52 Leaving Room)
09-03-2015_12-34-53.498 WARN [MOTION DETECTION EXTENSION]: Motion detected. -
09-03-2015_12-34-54.545 WARN [MOTION DETECTION EXTENSION]: Motion detected.
(12:34:55 Closing Door)
09-03-2015_12-34-55.149 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_12-34-55.791 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_12-34-58.483 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_12-34-58.487 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_12-34-58.495 INFO [Bluetooth DD]: Untrustworthy device discovered.
(12:34:59 Behind Door)

(01:53 START MARK 4 / Behind Door)
```

```
09-03-2015_12-35-03.557 INFO [Noise Detection]: Noise detected (60).
09-03-2015_12-35-03.686 INFO [Noise Detection]: Noise detected (58).
(12:35:05 Knocking Door)
09-03-2015_12-35-07.641 INFO [Noise Detection]: Noise detected (57). (JA)
(12:35:08 Opening Door) + (12:35:09 Inside Room) = Entering Room
(12:35:11 Speaking START*)
09-03-2015_12-35-13.448 WARN [MOTION DETECTION EXTENSION]: Motion detected. -
09-03-2015_12-35-13.880 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_12-35-14.133 INFO [Noise Detection]: Noise detected (55).
09-03-2015_12-35-14.481 WARN [MOTION DETECTION EXTENSION]: Motion detected.
(*12:35:15 Speaking END)
09-03-2015_12-35-15.165 WARN [MOTION DETECTION EXTENSION]: Motion detected. -
09-03-2015_12-35-19.749 WARN [MOTION DETECTION EXTENSION]: Motion detected.
(12:35:20 Leaving Room)
09-03-2015_12-35-20.241 WARN [MOTION DETECTION EXTENSION]: Motion detected. -
09-03-2015_12-35-23.441 WARN [MOTION DETECTION EXTENSION]: Motion detected.
(12:35:25 Closing Door)
(12:35:29 Behind Door)
09-03-2015_12-35-40.072 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_12-35-40.077 INFO [Bluetooth DD]: Untrustworthy device discovered.

(02:34 START Screen in FOV / Behind Door)
09-03-2015_12-35-43.738 INFO [Noise Detection]: Noise detected (56).
09-03-2015_12-35-43.866 INFO [Noise Detection]: Noise detected (69).
09-03-2015_12-35-43.993 INFO [Noise Detection]: Noise detected (56).
(12:35:44 Knocking Door)
09-03-2015_12-35-47.139 INFO [Noise Detection]: Noise detected (55). (JA)
09-03-2015_12-35-47.267 INFO [Noise Detection]: Noise detected (58).
(12:35:47 Opening Door) + (12:35:48 Inside Room) = Entering Room
09-03-2015_12-35-51.851 WARN [MOTION DETECTION EXTENSION]: Motion detected. -
09-03-2015_12-35-53.045 WARN [MOTION DETECTION EXTENSION]: Motion detected.
(12:35:54 Screen in the field of vision)
09-03-2015_12-36-04.573 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_12-36-04.579 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_12-36-04.590 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_12-36-05.837 WARN [Face Detection]: 2 face(s) detected.
09-03-2015_12-36-14.156 INFO [Face Detection]: 1 face(s) detected.
(12:36:26 Leaving Room)
09-03-2015_12-36-30.774 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_12-36-30.778 WARN [MOTION DETECTION EXTENSION]: Motion detected.
(12:36:32 Closing Door)
(12:36:35 Behind Door)
```

Listing 7.10: Evaluation Sheet Scenario 3

```
(00:00 START MARK 1 / Behind Door)
09-03-2015_13-28-05.077 INFO [Noise Detection]: Noise detected (57).
09-03-2015_13-28-05.214 INFO [Noise Detection]: Noise detected (55).
(13:28:07 Knocking Door 1) (JA)
(13:28:08 Opening Door 1)
09-03-2015_13-28-08.590 WARN [Motion Detection]: motion detected. -
09-03-2015_13-28-11.599 WARN [Motion Detection]: motion detected.
09-03-2015_13-28-11.599 INFO [Face Detection]: 1 face(s) detected.
09-03-2015_13-28-11.852 WARN [Motion Detection]: motion detected.
(13:28:12 Inside Room 1)
09-03-2015_13-28-12.122 WARN [Motion Detection]: motion detected. -
09-03-2015_13-28-12.834 WARN [Motion Detection]: motion detected.
(13:28:13 Speaking 1)
09-03-2015_13-28-13.481 WARN [Motion Detection]: motion detected.
(13:28:14 Knocking Door 2)
09-03-2015_13-28-15.169 WARN [Motion Detection]: motion detected.
09-03-2015_13-28-16.811 INFO [Noise Detection]: Noise detected (55). (JA)
09-03-2015_13-28-17.270 WARN [Motion Detection]: motion detected.
09-03-2015_13-28-17.531 WARN [Motion Detection]: motion detected.
(13:28:18 Opening Door 2) + (13:28:19 Inside Room 2) = Entering Room
(13:28:20 - 13:28:21 Speaking 2)
(13:28:23 Leaving Room & Closing Door 1+2)
09-03-2015_13-28-23.074 WARN [Motion Detection]: motion detected. -
09-03-2015_13-28-24.710 WARN [Motion Detection]: motion detected.
09-03-2015_13-28-25.045 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_13-28-25.049 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_13-28-25.006 WARN [Motion Detection]: motion detected. -
09-03-2015_13-28-26.545 WARN [Motion Detection]: motion detected.
```

```
09-03-2015_13-28-28.855 INFO [Face Detection]: 0 face(s) detected.
(13:28:28 Behind Door 1+2)

(00:28 START MARK 2 / Behind Door)
09-03-2015_13-28-33.375 INFO [Noise Detection]: Noise detected (55).
(13:28:34 Knocking Door 1)
09-03-2015_13-28-35.516 INFO [Noise Detection]: Noise detected (55). (JA)
(13:28:37 Opening Door 1)
09-03-2015_13-28-37.061 WARN [Motion Detection]: motion detected. -
09-03-2015_13-28-37.999 WARN [Motion Detection]: motion detected.
(13:28:38 Inside Room 1)
09-03-2015_13-28-38.252 WARN [Motion Detection]: motion detected. -
09-03-2015_13-28-39.876 WARN [Motion Detection]: motion detected.
(13:28:40 Speaking 1)
09-03-2015_13-28-40.111 WARN [Motion Detection]: motion detected.
09-03-2015_13-28-40.385 WARN [Motion Detection]: motion detected.
09-03-2015_13-28-40.385 INFO [Face Detection]: 1 face(s) detected.
09-03-2015_13-28-40.642 WARN [Motion Detection]: motion detected. -
09-03-2015_13-28-43.523 WARN [Motion Detection]: motion detected.
(13:28:44 Knocking Door 2) (JA)
09-03-2015_13-28-44.184 WARN [Motion Detection]: motion detected. -
09-03-2015_13-28-44.616 WARN [Motion Detection]: motion detected.
(13:28:45 Opening Door 2)
09-03-2015_13-28-46.060 WARN [Motion Detection]: motion detected.
09-03-2015_13-28-46.304 WARN [Motion Detection]: motion detected.
(13:28:47 Inside Room 2)
09-03-2015_13-28-48.892 WARN [Motion Detection]: motion detected.
(13:28:49 Speaking 2)
09-03-2015_13-28-49.241 WARN [Motion Detection]: motion detected. -
09-03-2015_13-28-49.743 WARN [Motion Detection]: motion detected.
09-03-2015_13-28-49.921 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_13-28-49.925 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_13-28-51.176 WARN [Motion Detection]: motion detected.
09-03-2015_13-28-51.899 WARN [Motion Detection]: motion detected.
(13:28:52 Leaving Room 1+2)
09-03-2015_13-28-52.140 WARN [Motion Detection]: motion detected. -
09-03-2015_13-28-53.869 WARN [Motion Detection]: motion detected.
(13:28:54 Closing Door 1+2)
09-03-2015_13-28-54.162 WARN [Motion Detection]: motion detected. -
09-03-2015_13-28-54.645 WARN [Motion Detection]: motion detected.
09-03-2015_13-28-54.886 INFO [Face Detection]: 0 face(s) detected.
09-03-2015_13-28-54.886 WARN [Motion Detection]: motion detected. -
09-03-2015_13-28-56.799 WARN [Motion Detection]: motion detected.
09-03-2015_13-28-57.214 INFO [Face Detection]: 1 face(s) detected.
09-03-2015_13-28-58.991 INFO [Noise Detection]: Noise detected (55).
09-03-2015_13-28-59.563 INFO [Face Detection]: 0 face(s) detected.
(13:28:59 Behind Door 1+2)

(01:10 START MARK 3 / Behind Door)
09-03-2015_13-29-15.018 INFO [Noise Detection]: Noise detected (62).
09-03-2015_13-29-15.148 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_13-29-15.152 INFO [Bluetooth DD]: Untrustworthy device discovered.
(13:29:16 Knocking Door 1)
09-03-2015_13-29-17.447 INFO [Noise Detection]: Noise detected (55). (JA)
(13:29:18 Opening Door 1)
09-03-2015_13-29-18.623 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-19.714 WARN [Motion Detection]: motion detected.
(13:29:20 Inside Room 1)
09-03-2015_13-29-20.000 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-21.576 WARN [Motion Detection]: motion detected.
09-03-2015_13-29-21.576 INFO [Face Detection]: 1 face(s) detected.
09-03-2015_13-29-21.876 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-22.930 WARN [Motion Detection]: motion detected.
(13:29:23 Speaking 1)
09-03-2015_13-29-23.169 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-26.754 WARN [Motion Detection]: motion detected.
(13:29:27 Knocking Door 2)
09-03-2015_13-29-27.056 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-28.593 WARN [Motion Detection]: motion detected.
09-03-2015_13-29-28.826 INFO [Noise Detection]: Noise detected (55). (JA)
09-03-2015_13-29-28.859 WARN [Motion Detection]: motion detected.
09-03-2015_13-29-29.127 WARN [Motion Detection]: motion detected.
(13:29:30 Opening Door 2)
```

```
09-03-2015_13-29-30.126 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-30.977 WARN [Motion Detection]: motion detected.
(13:29:31 Inside Room 2)
09-03-2015_13-29-31.224 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-33.288 WARN [Motion Detection]: motion detected.
(13:29:34 Speaking 2)
09-03-2015_13-29-34.334 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_13-29-34.719 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-36.816 WARN [Motion Detection]: motion detected.
(13:29:37 Leaving Room 1+2)
09-03-2015_13-29-37.149 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-38.537 WARN [Motion Detection]: motion detected.
09-03-2015_13-29-38.537 INFO [Face Detection]: 0 face(s) detected.
09-03-2015_13-29-38.863 WARN [Motion Detection]: motion detected.
09-03-2015_13-29-39.006 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_13-29-39.247 WARN [Motion Detection]: motion detected.
09-03-2015_13-29-39.611 WARN [Motion Detection]: motion detected.
09-03-2015_13-29-39.906 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_13-29-39.912 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_13-29-39.924 WARN [Motion Detection]: motion detected.
(13:29:40 Closing Door 1+2)
09-03-2015_13-29-40.286 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-41.367 WARN [Motion Detection]: motion detected.
09-03-2015_13-29-41.367 INFO [Face Detection]: 1 face(s) detected.
09-03-2015_13-29-42.003 WARN [Motion Detection]: motion detected.
09-03-2015_13-29-42.310 WARN [Motion Detection]: motion detected.
09-03-2015_13-29-43.212 INFO [Face Detection]: 0 face(s) detected.
(13:29:43 Behind Door 1+2)

(01:48 START MARK 4 / Behind Door)
09-03-2015_13-29-53.187 INFO [Noise Detection]: Noise detected (57).
(13:29:54 Knocking Door 1)
09-03-2015_13-29-55.602 INFO [Noise Detection]: Noise detected (55). (JA)
(13:29:56 Opening Door 1)
09-03-2015_13-29-56.689 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-57.911 WARN [Motion Detection]: motion detected.
(13:29:58 Inside Room 1)
09-03-2015_13-29-58.226 WARN [Motion Detection]: motion detected. -
09-03-2015_13-29-59.296 WARN [Motion Detection]: motion detected.
09-03-2015_13-29-59.642 INFO [Face Detection]: 1 face(s) detected.
09-03-2015_13-29-59.642 WARN [Motion Detection]: motion detected. -
09-03-2015_13-30-01.655 WARN [Motion Detection]: motion detected.
(13:30:02 Speaking 1)
09-03-2015_13-30-02.025 WARN [Motion Detection]: motion detected.
09-03-2015_13-30-02.025 INFO [Face Detection]: 0 face(s) detected.
09-03-2015_13-30-02.312 WARN [Motion Detection]: motion detected. -
09-03-2015_13-30-04.117 WARN [Motion Detection]: motion detected.
09-03-2015_13-30-04.566 INFO [Bluetooth DD]: Trusted device discovered.
09-03-2015_13-30-04.573 INFO [Bluetooth DD]: Untrustworthy device discovered.
09-03-2015_13-30-04.757 WARN [Motion Detection]: motion detected. -
09-03-2015_13-30-05.490 WARN [Motion Detection]: motion detected.
(13:30:06 Knocking Door 2) (JA)
09-03-2015_13-30-06.011 WARN [Motion Detection]: motion detected. -
09-03-2015_13-30-08.891 WARN [Motion Detection]: motion detected.
(13:30:09 Opening Door 2)
09-03-2015_13-30-09.236 WARN [Motion Detection]: motion detected. -
09-03-2015_13-30-09.744 WARN [Motion Detection]: motion detected.
(13:30:10 Inside Room 2)
09-03-2015_13-30-10.127 WARN [Motion Detection]: motion detected.
09-03-2015_13-30-10.602 WARN [Motion Detection]: motion detected.
09-03-2015_13-30-10.743 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_13-30-10.935 WARN [Motion Detection]: motion detected. -
09-03-2015_13-30-12.477 WARN [Motion Detection]: motion detected.
09-03-2015_13-30-12.928 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_13-30-12.865 WARN [Motion Detection]: motion detected.
(13:30:13 Speaking 2)
09-03-2015_13-30-13.232 WARN [Motion Detection]: motion detected.
09-03-2015_13-30-13.589 WARN [Motion Detection]: motion detected.
09-03-2015_13-30-14.053 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_13-30-14.263 WARN [Motion Detection]: motion detected.
09-03-2015_13-30-14.529 WARN [Motion Detection]: motion detected.
09-03-2015_13-30-14.609 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09-03-2015_13-30-14.733 WARN [Motion Detection]: motion detected. -
```

```
09−03−2015_13−30−16.891 WARN [Motion Detection]: motion detected.
(13:30:17 Leaving Room 1+2)
09−03−2015_13−30−17.194 WARN [Motion Detection]: motion detected. −
09−03−2015_13−30−18.313 WARN [Motion Detection]: motion detected.
09−03−2015_13−30−18.693 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09−03−2015_13−30−18.832 WARN [Motion Detection]: motion detected.
09−03−2015_13−30−19.177 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09−03−2015_13−30−19.266 WARN [Motion Detection]: motion detected.
09−03−2015_13−30−19.686 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09−03−2015_13−30−19.729 WARN [Motion Detection]: motion detected.
09−03−2015_13−30−20.203 WARN [Motion Detection]: motion detected.
09−03−2015_13−30−20.366 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09−03−2015_13−30−20.478 WARN [Motion Detection]: motion detected.
09−03−2015_13−30−20.817 WARN [Motion Detection]: motion detected.
(13:30:21 Closing Door 1+2)
09−03−2015_13−30−21.160 WARN [Motion Detection]: motion detected.
09−03−2015_13−30−21.409 WARN [Motion Detection]: motion detected.
09−03−2015_13−30−21.496 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09−03−2015_13−30−21.772 WARN [Motion Detection]: motion detected.
09−03−2015_13−30−22.053 WARN [MOTION DETECTION EXTENSION]: Motion detected.
09−03−2015_13−30−22.189 WARN [Motion Detection]: motion detected.
09−03−2015_13−30−22.830 INFO [Face Detection]: 1 face(s) detected.
09−03−2015_13−30−22.830 WARN [Motion Detection]: motion detected. −
09−03−2015_13−30−24.031 WARN [Motion Detection]: motion detected.
09−03−2015_13−30−24.031 INFO [Face Detection]: 0 face(s) detected.
(Behind Door 1+2)
```

Bibliography

[ACLT00] Roland Auckenthaler, Michael Carey, and Harvey Lloyd-Thomas. Score normalization for text-independent speaker verification systems. *Digital Signal Processing*, 10(1):42–54, 2000.

[AS01] Douglas Ayers and Mubarak Shah. Monitoring human behavior from video taken in an office environment. *Image and Vision Computing*, 19(12):833–846, 2001.

[BJE$^+$08] Yannick Benezeth, P-M Jodoin, Bruno Emile, Hélène Laurent, and Christophe Rosenberger. Review and evaluation of commonly-implemented background subtraction algorithms. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.

[BLGT06] Manuele Bicego, Andrea Lagorio, Enrico Grosso, and Massimo Tistarelli. On the use of sift features for face authentication. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*, pages 35–35. IEEE, 2006.

[BOHS14] Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. Ten years of pedestrian detection, what have we learned? In *Computer Vision-ECCV 2014 Workshops*, pages 613–627. Springer, 2014.

[Bur87] David K Burton. Text-dependent speaker verification using vector quantization source coding. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(2):133–143, 1987.

[CCR$^+$03] William M Campbell, Joseph P Campbell, Douglas A Reynolds, Douglas A Jones, and Timothy R Leek. Phonetic speaker recognition with support vector machines. In *Advances in neural information processing systems*, page None, 2003.

[CCR$^+$06] William M Campbell, Joseph P Campbell, Douglas A Reynolds, Elliot Singer, and Pedro A Torres-Carrasquillo. Support vector machines for speaker and language recognition. *Computer Speech & Language*, 20(2):210–229, 2006.

[CER05] Chloé Clavel, Thibaut Ehrette, and Gaël Richard. Events detection for an audio-based surveillance system. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 1306–1309. IEEE, 2005.

[CGPP03] Rita Cucchiara, Costantino Grana, Massimo Piccardi, and Andrea Prati. Detecting moving objects, ghosts, and shadows in video streams. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(10):1337–1342, 2003.

[CHZ$^+$03] Longbin Chen, Baogang Hu, Lei Zhang, Mingjing Li, and HongJiang Zhang. Face annotation for family photo album management. *International Journal of Image and Graphics*, 3(01):81–94, 2003.

[CJ97] Joseph P Campbell Jr. Speaker recognition: A tutorial. *Proceedings of the IEEE*, 85(9):1437–1462, 1997.

[CLS$^+$01] Michael J Covington, Wende Long, Srividhya Srinivasan, Anind K Dev, Mustaque Ahamad, and Gregory D Abowd. Securing context-aware applications using environment roles. In *Proceedings of the sixth ACM symposium on Access control models and technologies*, pages 10–20. ACM, 2001.

[CSR06] William M Campbell, Douglas E Sturim, and Douglas A Reynolds. Support vector machines using gmm supervectors for speaker verification. *Signal Processing Letters, IEEE*, 13(5):308–311, 2006.

[CTB92] Ian Craw, David Tock, and Alan Bennett. Finding face features. In *Computer VisionâĂŤECCV'92*, pages 92–96. Springer, 1992.

[DFB99] Benoit Duc, Stefan Fischer, and Josef Bigün. Face authentication with gabor information on deformable graphs. *Image Processing, IEEE Transactions on*, 8(4):504–516, 1999.

[DGFVG12] Matthias Dantone, Juergen Gall, Gabriele Fanelli, and Luc Van Gool. Real-time facial feature detection using conditional regression forests. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2578–2585. IEEE, 2012.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[DWSP09] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 304–311. IEEE, 2009.

[EBT$^+$09] José Port Elo, Miguel Bugalho, Isabel Trancoso, Joao Neto, Alberto Abad, and Antonio Serralheiro. Non-speech audio event detection. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 1973–1976. IEEE, 2009.

[EHD00] Ahmed Elgammal, David Harwood, and Larry Davis. Non-parametric model for background subtraction. In *Computer VisionâĂŤECCV 2000*, pages 751–767. Springer, 2000.

[Far95] Kevin R Farrell. Text-dependent speaker verification using data fusion. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 349–352. IEEE, 1995.

[FCSB89] DK Freeman, G Cosier, CB Southcott, and I Boyd. The voice activity detector for the pan-european digital cellular mobile telephone service. In *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, pages 369–372. IEEE, 1989.

[FK09] David F Ferraiolo and D Richard Kuhn. Role-based access controls. *arXiv preprint arXiv:0903.2171*, 2009.

[FKC03] David Ferraiolo, D Richard Kuhn, and Ramaswamy Chandramouli. *Role-based access control*. Artech House, 2003.

[FKCB92] David F Ferraiolo, D Richard Kuhn, Ramaswamy Chandramouli, and John Barkley. Role-based access control (rbac). In *15th National Computer Security Conference*, pages 554–563, 1992.

[FL03] Beat Fasel and Juergen Luettin. Automatic facial expression analysis: a survey. *Pattern recognition*, 36(1):259–275, 2003.

[FPS11] Ludwig Fuchs, Günther Pernul, and Ravi Sandhu. Roles in information security–a survey and classification of the research area. *computers & security*, 30(8):748–769, 2011.

[fSG89] International Organization for Standardization (Gèneve). *Information Processing Systems: Open Systems Interconnection: LOTOS: a Formal Description Technique Based on the Temporal Ordening of Observational Behaviour*. International Organization for Standardization, 1989.

[FSG$^+$01] David F Ferraiolo, Ravi Sandhu, Serban Gavrila, D Richard Kuhn, and Ramaswamy Chandramouli. Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274, 2001.

[G$^+$09] OM Group et al. Omg unified modeling language (omg uml), superstructure. *Open Management Group*, 2009.

[GBS$^+$13a] Dimitrios Giannoulis, Emmanouil Benetos, Dan Stowell, Mathias Rossignol, Mathieu Lagrange, and Mark Plumbley. Detection and classification of acoustic scenes and events. *an IEEE AASP Challenge*, 2013.

[GBS$^+$13b] Dimitrios Giannoulis, Emmanouil Benetos, Dan Stowell, Mathias Rossignol, Mathieu Lagrange, and Mark D Plumbley. Detection and classification of acoustic scenes and events: An ieee aasp challenge. In *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2013 IEEE Workshop on*, pages 1–4. IEEE, 2013.

[GCZ$^+$13] Xincheng Gao, Houbin Cao, Jianfeng Zhang, Jinping Bai, Tianhang Zhang, and Lihong Jia. A real-time dsp-based system for voice activity detection: Design and implement. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 6(6):27–40, 2013.

[GSB$^+$13] Dimitrios Giannoulis, Dan Stowell, Emmanouil Benetos, Mathias Rossignol, Mathieu Lagrange, and Mark D Plumbley. A database and challenge for acoustic scene classification and event detection. In *Signal Processing Conference (EUSIPCO), 2013 Proceedings of the 21st European*, pages 1–5. IEEE, 2013.

[Har87] David Harel. Statecharts: A visual formalism for complex systems. *Science of computer programming*, 8(3):231–274, 1987.

[HFK$^+$14] Vincent C Hu, David Ferraiolo, Rick Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, and Karen Scarfone. Guide to attribute based access control (abac) definition and considerations. *NIST Special Publication*, 800:162, 2014.

[HHD99] Thanarat Horprasert, David Harwood, and Larry S Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *IEEE ICCV*, volume 99, pages 1–19, 1999.

[HSXS13] Jungong Han, Ling Shao, Dong Xu, and Jamie Shotton. Enhanced computer vision with microsoft kinect sensor: A review. *Cybernetics, IEEE Transactions on*, 43(5):1318–1334, 2013.

[Hur89] Anya C Hurlbert. The computation of color. Technical report, DTIC Document, 1989.

[IM97] Kenzo Itoh and Masahide Mizushima. Environmental noise reduction based on speech/non-speech identification for hearing aids. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 1, pages 419–422. IEEE, 1997.

[KCT00] Takeo Kanade, Jeffrey F Cohn, and Yingli Tian. Comprehensive database for facial expression analysis. In *Automatic Face and Gesture*

Recognition, 2000. Proceedings. Fourth IEEE International Conference on, pages 46–53. IEEE, 2000.

[KHK08] Takuya Kobayashi, Akinori Hidaka, and Takio Kurita. Selection of histograms of oriented gradients features for pedestrian detection. In *Neural Information Processing*, pages 598–607. Springer, 2008.

[KK96] Rick Kjeldsen and John Kender. Finding skin in color images. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pages 312–317. IEEE, 1996.

[KL10] Tomi Kinnunen and Haizhou Li. An overview of text-independent speaker recognition: From features to supervectors. *Speech communication*, 52(1):12–40, 2010.

[KPS03] Jure Kovač, Peter Peer, and Franc Solina. *Human skin color clustering for face detection*, volume 2. IEEE, 2003.

[KT08] Devdatta Kulkarni and Anand Tripathi. Context-aware role-based access control in pervasive computing systems. In *Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 113–122. ACM, 2008.

[KWSN96] Yuji Kuno, Takahiro Watanabe, Yoshinori Shimosakoda, and Satoshi Nakagawa. Automated detection of human for visual surveillance system. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 3, pages 865–869. IEEE, 1996.

[Lam69] Butler W Lampson. Dynamic protection structures. In *Proceedings of the November 18-20, 1969, fall joint computer conference*, pages 27–38. ACM, 1969.

[Lam74] Butler W Lampson. Protection. *ACM SIGOPS Operating Systems Review*, 8(1):18–24, 1974.

[LBP95] Thomas K Leung, Michael C Burl, and Pietro Perona. Finding faces in cluttered scenes using random labeled graph matching. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 637–644. IEEE, 1995.

[Lin06] Hakan Lindqvist. Mandatory access control. *Master's Thesis in Computing Science, Umea University, Department of Computing Science, SE-901*, 87, 2006.

[LSS05] Bastian Leibe, Edgar Seemann, and Bernt Schiele. Pedestrian detection in crowded scenes. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 878–885. IEEE, 2005.

[LV01] BPL Lo and SA Velastin. Automatic congestion detection system for underground platforms. In *Intelligent Multimedia, Video and Speech Processing, 2001. Proceedings of 2001 International Symposium on*, pages 158–161. IEEE, 2001.

[Mal89] Stephane G Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(7):674–693, 1989.

[MCT09] Erik Murphy-Chutorian and Mohan Manubhai Trivedi. Head pose estimation in computer vision: A survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(4):607–626, 2009.

[MGR98] Stephen J McKenna, Shaogang Gong, and Yogesh Raja. Modelling facial colour and identity with gaussian mixtures. *Pattern recognition*, 31(12):1883–1892, 1998.

[MN13] Yanna Ma and Akinori Nishihara. Efficient voice activity detection algorithm using long-term spectral flatness measure. *EURASIP Journal on Audio, Speech, and Music Processing*, 2013(1):1–18, 2013.

[MSZ04] Krystian Mikolajczyk, Cordelia Schmid, and Andrew Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *Computer Vision-ECCV 2004*, pages 69–82. Springer, 2004.

[MT03] Emmanuel Munguia Tapia. *Activity recognition in the home setting using simple and ubiquitous sensors.* PhD thesis, Massachusetts Institute of Technology, 2003.

[Nie07] Eva Nieuwdorp. The pervasive discourse: an analysis. *Computers in Entertainment (CIE)*, 5(2):13, 2007.

[OFG97] Edgar Osuna, Robert Freund, and Federico Girosi. Training support vector machines: an application to face detection. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 130–136. IEEE, 1997.

[OPS$^+$97] Michael Oren, Constantine Papageorgiou, Pawan Sinha, Edgar Osuna, and Tomaso Poggio. Pedestrian detection using wavelet templates. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 193–199. IEEE, 1997.

[Pic04] Massimo Piccardi. Background subtraction techniques: a review. In *Systems, man and cybernetics, 2004 IEEE international conference on*, volume 4, pages 3099–3104. IEEE, 2004.

[PMR+00] P Jonathon Phillips, Hyeonjoon Moon, Syed Rizvi, Patrick J Rauss, et al. The feret evaluation methodology for face-recognition algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(10):1090–1104, 2000.

[POP98] Constantine P Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *Computer vision, 1998. sixth international conference on*, pages 555–562. IEEE, 1998.

[PP99] Constantine Papageorgiou and Tomaso Poggio. Trainable pedestrian detection. In *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, volume 4, pages 35–39. IEEE, 1999.

[PPT+11] Georgios Passalis, Panagiotis Perakis, Theoharis Theoharis, Ioannis Kakadiaris, et al. Using facial symmetry to handle pose variations in real-world 3d face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(10):1938–1951, 2011.

[PT03] Fatih Porikli and Oncel Tuzel. Human body tracking by adaptive background models and mean-shift analysis. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 1–9. Citeseer, 2003.

[QZW+85] Lili Qiu, Yin Zhang, Feng Wang, Mi Kyung, and Han Ratul Mahajan. Trusted computer system evaluation criteria. In *National Computer Security Center*. Citeseer, 1985.

[RFY+13] Stephen N Robinovitch, Fabio Feldman, Yijian Yang, Rebecca Schonnop, Pet Ming Leung, Thiago Sarraf, Joanie Sims-Gould, and Marie Loughin. Video capture of the circumstances of falls in elderly people residing in long-term care: an observational study. *The Lancet*, 381(9860):47–54, 2013.

[RGS07] Javier Ramirez, Juan Manuel Górriz, and José Carlos Segura. *Voice activity detection. fundamentals and speech recognition system robustness*. INTECH Open Access Publisher, 2007.

[RJ98] Thomas D Rikert and Michael J Jones. Gaze estimation using morphable models. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, pages 436–441. IEEE, 1998.

[RK08] Mohammad T Rahman and Nasser Kehtarnavaz. Real-time face-priority auto focus for digital and cell-phone cameras. *Consumer Electronics, IEEE Transactions on*, 54(4):1506–1513, 2008.

[RQD00] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn. Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1):19–41, 2000.

[RR⁺90] Richard C Rose, Douglas Reynolds, et al. Text independent speaker identification using automatic acoustic segmentation. In *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pages 293–296. IEEE, 1990.

[RR⁺95] Douglas Reynolds, Richard C Rose, et al. Robust text-independent speaker identification using gaussian mixture speaker models. *Speech and Audio Processing, IEEE Transactions on*, 3(1):72–83, 1995.

[RS98] Romer Rosales and Stan Sclaroff. Improved tracking of multiple humans with trajectory prediction and occlusion modeling. In *IEEE CVPR workshop on the Interpretation of Visual Motion*, 1998.

[RSB⁺04] Javier Ramırez, José C Segura, Carmen Benıtez, Angel De La Torre, and Antonio Rubio. Efficient voice activity detection algorithms using long-term speech information. *Speech communication*, 42(3):271–287, 2004.

[San98] Ravi S Sandhu. Role-based access control. *Advances in computers*, 46:237–286, 1998.

[SB08] William Stallings and Lawrie Brown. Computer security. *Principles and Practice*, 2008.

[SCFY96] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, (2):38–47, 1996.

[SFK00] Ravi Sandhu, David Ferraiolo, and Richard Kuhn. The nist model for role-based access control: towards a unified standard. In *ACM workshop on Role-based access control*, volume 2000, 2000.

[SG99] Chris Stauffer and W Eric L Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999.

[SGH04] Amnon Shashua, Yoram Gdalyahu, and Gaby Hayun. Pedestrian detection for driving assistance systems: Single-frame classification and system level performance. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 1–6. IEEE, 2004.

[SKK04] Marios Savvides, BVKV Kumar, and Pradeep K Khosla. Cancelable biometric filters for face recognition. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 922–925. IEEE, 2004.

[SKS99] Jongseo Sohn, Nam Soo Kim, and Wonyong Sung. A statistical model-based voice activity detection. *Signal Processing Letters, IEEE*, 6(1):1–3, 1999.

[SS94] Ravi S Sandhu and Pierangela Samarati. Access control: principle and practice. *Communications Magazine, IEEE*, 32(9):40–48, 1994.

[ST01] Cristian Sminchisescu and Bill Triggs. Covariance scaled sampling for monocular 3d body tracking. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–447. IEEE, 2001.

[Tho04] Tim Thornburgh. Social engineering: the dark art. In *Proceedings of the 1st annual conference on Information security curriculum development*, pages 133–135. ACM, 2004.

[TMKL06] Alessandra Toninelli, Rebecca Montanari, Lalana Kagal, and Ora Lassila. A semantic context-aware access control framework for secure collaborations in pervasive computing environments. In *The Semantic Web-ISWC 2006*, pages 473–486. Springer, 2006.

[TP91a] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[TP$^+$91b] Matthew Turk, Alex P Pentland, et al. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591. IEEE, 1991.

[UFF06] Raquel Urtasun, David J Fleet, and Pascal Fua. Temporal motion models for monocular and multiview 3d human body tracking. *Computer vision and image understanding*, 104(2):157–177, 2006.

[VJ01] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[VSG12] Roberto Valenti, Nicu Sebe, and Theo Gevers. Combining head pose and eye location information for gaze estimation. *Image Processing, IEEE Transactions on*, 21(2):802–815, 2012.

[VSSG09] Roberto Valenti, Jacopo Staiano, Nicu Sebe, and Theo Gevers. Webcam-based visual gaze estimation. In *Image Analysis and Processing–ICIAP 2009*, pages 662–671. Springer, 2009.

[WADP97] Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Paul Pentland. Pfinder: Real-time tracking of the human body. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):780–785, 1997.

[WF06] Phillip Ian Wilson and John Fernandez. Facial feature detection using haar classifiers. *Journal of Computing Sciences in Colleges*, 21(4):127–133, 2006.

[WL$^+$05] Christopher Waring, Xiuwen Liu, et al. Face detection using spectral histograms and svms. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 35(3):467–476, 2005.

[WN05] Bo Wu and Ram Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 90–97. IEEE, 2005.

[Wor07] Michael Workman. Gaining access with social engineering: An empirical study of the threat. *Information Systems Security*, 16(6):315–331, 2007.

[WSV03] Jiangang Wang, Eric Sung, and Ronda Venkateswarlu. Eye gaze estimation from a single image of one eye. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 136–143. IEEE, 2003.

[XRDH03] Ziyou Xiong, Regunathan Radhakrishnan, Ajay Divakaran, and Thomas S Huang. Audio events detection based highlights extraction from baseball, golf and soccer games in a unified framework. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 5, pages V–632. IEEE, 2003.

[YC05] Dong Hyun Yoo and Myung Jin Chung. A novel non-intrusive eye gaze estimation using cross-ratio under large head motion. *Computer Vision and Image Understanding*, 98(1):25–51, 2005.

[YH94] Guangzheng Yang and Thomas S Huang. Human face detection in a complex background. *Pattern recognition*, 27(1):53–63, 1994.

[YKA02] Ming-Hsuan Yang, David J Kriegman, and Narendra Ahuja. Detecting faces in images: A survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(1):34–58, 2002.

[YZ02] Ruigang Yang and Zhengyou Zhang. Model-based head pose tracking with stereovision. In *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, pages 255–260. IEEE, 2002.

[ZCPR03] Wenyi Zhao, Rama Chellappa, P Jonathon Phillips, and Azriel Rosenfeld. Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4):399–458, 2003.

[ZJ04] Zhiwei Zhu and Qiang Ji. Eye and gaze tracking for interactive graphic display. *Machine Vision and Applications*, 15(3):139–148, 2004.

[ZP04] Guangsen Zhang and Manish Parashar. Context-aware dynamic access control for pervasive applications. In *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference*, pages 21–30, 2004.

[ZYCA06] Qiang Zhu, M-C Yeh, Kwang-Ting Cheng, and Shai Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1491–1498. IEEE, 2006.

[ZZ05] Dongxiang Zhou and Hong Zhang. Modified gmm background modeling and optical flow for detection of moving objects. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 3, pages 2224–2229. IEEE, 2005.

[ZZ10] Cha Zhang and Zhengyou Zhang. A survey of recent advances in face detection. Technical report, Tech. rep., Microsoft Research, 2010.

List of Figures

List of Tables

List of Algorithms

List of Listings

Acronyms

1-G One Gaussian. 26

ABAC Attribute-based Access Control. 11, 13, 14, 19, 94
ACLs Access Control Lists. 7, 8, 17, 93
API Application Programming Interface. 48
ASI Automatic Speaker Identification. 31
ASV Automatic Speaker Verification. 31

CHMs Cylindrical Head Models. 28

DAC Discretionary Access Control. 6, 7, 9–11, 13, 17, 94
DFT Discrete Fourier Transform. 31

FFT Fast Fourier Transform. 31

GMM Gaussian Mixture Model. 26, 27, 33
GRNNs Generalized Regression Neural Networks. 28
GUI Graphical User Interface. 42

HCI Human-Computer Interaction. 24
HMM Hidden Markov Models. 29–31
HOG Histogram of Oriented Gradients. 21, 22

KDE Kernel Density Estimation. 26

LRT Likelihood Ratio Test. 31
LSFM Long-Term Spectral Flatness Measure. 31

LTSD Long-Term Spectral Divergence. 31

LTSE Long-Term Spectral Envelope. 31

MAC Mandatory Access Control. 6, 9–11, 13, 17, 18, 94

MLS Multilevel Security. 9, 18

MPEG Moving Picture Experts Group. 29

MSF Root Mean Square Fluctuation. 48, 97

NIST National Institute of Standards and Technology. 11

PCA Principal Components Analysis. 22, 27

PDF Probability Distribution Function. 31

RBAC Role-based Access Control. 6, 9, 11–16, 18, 93, 94

RFID Radio-frequency Identification. 13

SNR Signal-to-Noise Ratio. 31

SVM Support Vector Machine. 21, 25, 30, 33

UML Unified Modeling Language. 57

VAD Voice Activity Detection. 30, 31

WLAN Wireless Local Area Network. 5