

Bridging Sensing and Decision Making in Ambient Intelligence Environments

Elie Raad, Bechara Al Bouna, Richard Chbeir

▶ **To cite this version:**

Elie Raad, Bechara Al Bouna, Richard Chbeir. Bridging Sensing and Decision Making in Ambient Intelligence Environments. Jeong, Jechang and Damiani, Ernesto. Multimedia Techniques for Device and Ambient Intelligence, Springer US, pp.135-164, 2009. hal-00643557

HAL Id: hal-00643557
https://hal.science/hal-00643557

Submitted on 17 Dec 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Chapter 6
Bridging Sensing and Decision Making in Ambient Intelligence Environments

Elie Raad, Bechara Al Bouna and Richard Chbeir

Summary. Context-aware and Ambient Intelligence environments represent one of the emerging issues in the last decade. In such intelligent environments, information is gathered to provide, on one hand, autonomic and easy to manage applications, and, on the other, secured access controlled environments. Several approaches have been defined in the literature to describe context-aware application with techniques to capture and represent information related to a specified domain. However and to the best of our knowledge, none has questioned the reliability of the techniques used to extract meaningful knowledge needed for decision making especially if the information captured is of multimedia types (images, sound, videos, etc.). In this chapter, we propose an approach to bridge the gap between sensing and decision making and provide an uncertainty resolver to reduce faulty decisions based on uncertain knowledge extracted from unreliable techniques. We describe also a set of experiments elaborated to demonstrate the efficiency of our uncertainty resolver.

Key words: context-aware application, semantic-based, uncertainty resolver model

Elie Raad
LE2I Laboratory UMR-CNRS, University of Bourgogne, 21078 Dijon Cedex France e-mail: elie.raad@u-bourgogne.fr

Bechara Al Bouna
LE2I Laboratory UMR-CNRS, University of Bourgogne, 21078 Dijon Cedex France e-mail: bechara.albouna@u-bourgogne.fr

Richard Chbeir
LE2I Laboratory UMR-CNRS, University of Bourgogne, 21078 Dijon Cedex France e-mail: richard.chbeir@u-bourgogne.fr

E. Damiani and J. Jeong (eds.), *Multimedia Techniques for Device and Ambient Intelligence*, DOI: 10.1007/978-0-387-88777-7_6,
© Springer Science + Business Media, LLC 2009

6.1 Introduction

Nowadays, ambient intelligence is receiving lot of attention in several application domains due to its faculty to provide controlled environments when equipped with multimedia sensors such as cameras, microphones, and others. In fact, with the use of smart devices and embedded sensors, systems are able to maintain relevant information about users. Multimedia data describing users reveal interesting information about their location and context (user surrounding, moves, gesture, etc.). The work done by the research community has led to the definition of several context-aware or pervasive approaches aiming at, on one hand, improving access control models [1] [2] [3] [4] [5] [6] and, on the other, assisting the user in performing their daily tasks [7] [8]. Context-aware approaches are designed with the purpose of defining applications capable of managing themselves without the direct intervention of users. They are characterized by their ability to guide users to perform appropriate tasks while providing reasoning and decisions about the actions to be triggered. Context awareness in ambient intelligence environments is considered one of the key issues of evolution toward a total and automatic computing paradigm. It allows a system to integrate human's ability in order to recognize and exploit implicit information related to users' surrounding. A context-aware system is viewed as a two layered framework separating the decision making from the sensed multimedia information. One of the challenging issues to handle in context-aware computing is how to bridge the gap between sensing and decision making. Let us consider the following motivating scenario of a company equipped with multimedia devices (surveillance cams, microphones, sensors, etc.) in each room and hallway to provide interacted environment for its employees. The company installed a central unit holding the decision making tool using a set of explicit information managed by the administrator. Each of the multimedia devices sends captured information to the central unit which in turn analyzes it and invokes the appropriate tasks. In the central unit, a set of multimedia functions is used to detect and recognize people and objects in a certain context. Thus, integrating multimedia data and functions (face recognition, image similarity, object recognition, etc.) in ambient intelligence environments may lead to frustrating situations and thus uncertain decisions to take. This comes from several assets (lightings, electrical noise, functions' relevance, etc.) and affects the related result. In order to validate a given fact, one should accept either a reasonable error risk or consider relying on several sources (e.g. taking several snapshots of an environment) using various multimedia functions enabling to retrieve the most appropriate result for a given situation. Consequently, we can easily pin down from this scenario two challenging issues:

1. How to allow an easy use of multimedia functions and the definition of new semantic-based functions after deriving a set of existing ones? This is of great importance to help the administrator to control his environment interaction and to evolve it accordingly when the company needs change.

2. How to allow reducing the risk of faulty decision according to a set of fuzzy inputs? This is indispensable when involving multimedia functions that provide fuzzy results when comparing objects or extracting features.

In this chapter, we address these two issues. We provide the concept of templates to facilitate the use of multimedia functions and derive new ones on the basis of combining existing ones. We also present here an uncertainty resolver model to reduce the potential risk of using multimedia functions in ambient intelligence environments. Our resolver is based on adaptations of known decision and probabilistic analysis techniques such as decisions trees [9], Bayesian networks [10], Dempster-Shafer theory [11], etc. to aggregate multimedia functions' results into one relevant computed result. Through a set of experimental tests, we show how our proposed approach can be beneficial and be tuned.
The rest of the chapter is defined as follows. In Section 2, we provide a quick overview on the context-aware models. In Section 3, we state the definitions needed to fully understand the proposed approach. Section 4 is devoted to describe the concept of template used in our approach to bridge the gap between the sensing and the decision making. A detailed description of our uncertainty resolver is given in Section 5 in which we provide a set of adapted aggregation functions used to reduce the uncertainty raised when processing multimedia objects. In Section 6, we demonstrate the efficiency related to integrating aggregation functions and their ability to reduce uncertainty and faulty decisions. Finally, we conclude the chapter and draw several perspectives.

6.2 Related Works

Context-aware systems have recently attracted a lot of attention and have been widely explored in the literature. Many projects have elaborated the context awareness as a key feature for pervasive computing. It has been mainly presented as an important aspect to enforce access control models. In this section, we focus only on presenting how current approaches (formally) describe the context information. We also snapshot major approaches integrating the context as a key issue to enforce related access control models. Context modeling and representation were initially based on the context widgets [12] used to separate the high level context modeling from the capturing level. This evolved later with the integration of ontologies as a way to represent context information. Ontologies proved to be useful in representing the semantics behind a given domain in which context is modeled as concepts and facts. It succeeded in a wide definition of high conceptualization layer and enforced this with automatic inferred and logic reasoning using explicit and implicit rules. For instance, in [13] the authors define a hybrid conceptual model combining relational modeling and ontology based modeling in order to represent contextual information. Whereas in [14], the authors provide an interesting approach able to treat high-level implicit contexts derived from low-level explicit context information. The approach is based on a generic context which could be extended to target

some domain specific concepts. In [15] and [16], both authors integrate contextual information in provided access control model to enforce the access decision and dynamically handle environmental changes in the user context. In [16], the authors provide an interesting model in which they incorporate the notion of trust. In this model, contextual information related to users' environment can affect the trust level assigned to the subject. Nevertheless, these models do not migrate well to handle multimedia contextual information. They lack the ability to define complex policies in which multimedia contextual information is evaluated. In [2] the authors define a policy based on context aware service for network environments. The policy model is based on predicates such as time, location, activity, etc. Further, in [6], a context-aware authorization model is defined to enhance security management for Intranet protocols. Context rules are defined here as constraints and integrated in a role based environment. In [4], context is semantically defined using ontologies and integrated into an access control model for pervasive computing environments. However, the previously described approaches are considered domain-oriented and application-dependent. They lack the ability to define complex policies integrating multimedia contextual conditions. In [17] and [18], the authors propose a Generalized Role Based Access Control model which extends the know RBAC model by incorporating the notion of environment roles, and object roles in addition to the know subject roles. The model, motivated by the Aware Home[1] security challenges, takes into consideration the information gathered from a variety of sensors and defines roles accordingly. Despite the efficiency provided by this model, it is commonly admitted that in large domains of application, the variety of roles is considered as a burden to handle by authorization managers. An interesting approach detailed in [19] describes a location-based access control able to authorize users on the basis of their location. Methods such as cell identification, angle of arrival, signal levels, etc. are used here to calculate the location position using GSM/3G devices and protocols. Nevertheless, such approach requires that each user (on which access should be controlled) carries a GSM/3G device which is not always feasible (for instance, in highly secured departments, users are forced to leave their devices at a security checkpoint). Furthermore, information acquired from cells is typically limited and endangered to noise and interference due to the wide areas the cells might cover. In [3], the authors enforce the role based access control model with context filters targeting context information. The proposed approach enforces, on one hand, the access to a subset of objects related to the user's context and, on the other hand, users' membership to a given role (or roles). However, the context filters represented in their approach are based on simple Boolean expression with logical and Boolean operators, and therefore they cannot be extended to deal with multimedia objects processing.

The approaches discussed here are based on context-aware applications where information representing a given surrounding is gathered to make platforms more autonomic, on one hand, and secure on the other. However, few have tackled the problem of reliability behind multimedia objects processing and the decisions to

[1] Aware Home is considered as a house with highly technical technology. It contains rich communication infrastructure in which sensors can capture and store a large size of information.

make in order to execute a given operation. In fact, due to the complex structure of multimedia data, extracting meaningful information from them is complex, fuzzy, uncertain, and time-consuming. In the following, we describe our approach allowing to bridge the gap between sensing and decision making and discuss our uncertainty resolver proposed to reduce uncertainty and faulty decisions.

6.3 Preliminaries

In the following, we present some definitions needed to fully understand our proposal.

Definition 1 - Multimedia Object (Mo): allows representing several types of multimedia data such as text, image, video, etc. It is formally represented in our approach as 4-Tuples of the following form:

$$Mo :< id, O, A, F > \tag{6.1}$$

where:

- **id**: represents the identifier of the multimedia object.
- **O**: contains the raw data of the object stored as a BLOB file or URI.
- **A**: is the metadata describing the multimedia object. It describes multimedia object related information and can be written as: (a1:v1, a2:v2) where ai and vi represent an attribute and its corresponding value (ex. age: 18, profession: student, etc.)
- ” **F**: describes the low-level features of a multimedia object (such as Color Histogram, Color distribution, Texture Histogram, shapes, Duration, Audio freq., Amplitude, Band no., etc.)

For instance, the following multimedia object:

$$\begin{aligned} Mo_1 : (id = 1, O =' photo_bob.jpg', A =' object.name : Bob', \\ F =' DominantColor = (14, 24, 20)') \end{aligned} \tag{6.2}$$

describes ”Bob” picture (the head of the research department in our motivating scenario) with its dominant color (using the RGB color space).

Definition 2 - Multimedia Function (f): is used to handle the comparison[2] and feature extraction of multimedia objects. Numerous types of multimedia functions are provided in several commercial tools and in the literature through various forms. For instance, several are provided in DBMSs SQL-operators such as Oracle and DB2 [20] [21] [22], while others are accessible via API functions [23] [24] and web

[2] When handling rules whose conditions include multimedia objects, traditional logical operators such as ’equality’, ’greater than’ or others are not applicable due to the complex structure of multimedia objects and must be extended with *similarity functions*.

service [25] for multimedia data processing. Details on such functions and their applications are out of the scope of this chapter. In our approach, we formally write a multimedia function f as:

$$f(Mo_j, Mo_i) \rightarrow \alpha^B f$$

where:

- Mo_jrepresents a predefined Multimedia Object
- Mo_i represents a captured multimedia object. It can be provided by multimedia devices of different types such as surveillance cameras, webcams, sound recorders, etc
- $\alpha^B f$ is a returned threshold representing the confidence score of a Boolean value B with respect to the multimedia function. It varies in $[01]interval$

For instance, consider the fact that we wish to detect the presence of Bob, head of the research department, in a snapshot image. It is possible to use 2 different multimedia functions f_1 and f_2 to analyze multimedia objects where:

- f_1: is related to the InterMedia Oracle module [22] and is used for image similarity,
- f_2: is based on color object recognition and an SVM classifier. It computes decisions based on a set of classes representing the trained images (See [26] for more details)

Thus, we can define the function contents as:

$$\begin{aligned} &f_1(Mo_1(\ldots 0 =' predBob.jpg' \ldots), Mo_{10}(\ldots, 0 =' snapshot_1.jpg', \ldots)) \rightarrow 0,5^T \\ &f_1(Mo_1(\ldots 0 =' predBob.jpg' \ldots), Mo_{10}(\ldots, 0 =' snapshot_1.jpg', \ldots)) \rightarrow 0,8^T \end{aligned} \tag{6.3}$$

where the predefined MO is a $predBob.jpg$ image[3] to be compared with an input object called $snapshot_1.jpg$ using the two multimedia functions. The first function detected the white suit with a 0.5 value whereas the second has 0.8 value.

Definition 3 - Multimedia Attribute Expression (MA): is a Boolean expression holding two set of multimedia objects as input for processing. It is formally described as:

$$M_A(MO_i, MO_j) = \mu(\{f_1(MO_k, MO_l), \ldots, f_n(MO_k, MO_l)\}, \{\varepsilon_1, \ldots, \varepsilon_n\})\theta \rightarrow \upsilon \tag{6.4}$$

where:

- $MO_i = Mo_1, \ldots, Mo_i$ and $MO_j = Mo_1, \ldots, Mo_j$ represent respectively the predefined set of multimedia objects and the captured set of multimedia objects to be compared with.
- μ is a aggregation function (to be detailed in Section 5) that holds a set of multimedia functions and uncertainty thresholds $\varepsilon_1, \ldots, \varepsilon_n$.

[3] Several functions require the signature of an image (e.g. dat files) during comparison. Here, we assume that signatures are generated on demand.

- f represents a multimedia comparison function needed to compare MO_k and $MO_l(MO_k \subseteq MO_i and MO_l \subseteq MO_j$.
- θ is a comparison operator containing traditional operators (e.g. $=, \neq, <, >, \leq, \geq$, etc.).
- υ is the validation score.

$MO_A(MO_i, MO_j)$ is satisfied if the result returned from the aggregation function compared to υ is valid. An example of using M_A will be provided later on.

6.4 Templates

The concept of template is used in our approach to put together common features needed to handle multimedia processing methods and functions. Templates provide, on one hand, ease of administration when using predefined templates for special use cases and, on the other hand, flexible manipulation of a set of multimedia functions to offer precision, time saving, and minimum uncertainty risks. They allow handling the analysis of a set of captured multimedia objects using multimedia functions and aggregation functions (to be described later). A template is formally described in our approach as follows:

$$T :< Id, Desc, M_A(MO_i, MO_j) >$$

where:

- Id represents the identifier of the template
- $Desc$ is the textual description of the template
- $M_A(MO_i, MO_j)$is a Boolean expression holding the multimedia attributes needed to process the set of multimedia objects MO_i and MO_j

For instance, consider the fact that we wish to detect (using the two different multimedia functions f_1 and f_2 defined earlier) the presence of Bob, head of the research department, in the conference room in order to alert all the employees in the office and make the environment adequate for a conference.

In order to provide simple manipulation and reduce uncertain decisions, we define a "Face Identification" template needed to combine both multimedia functions and aggregate their results using, for instance, an Average aggregate function Avg (detailed in the next section). Therefore, the template *Face Identification* can be defined as follows:

$$T_1 :< 001, "Faceidentification", M_A(Mo_1, Mo_{10}, Mo_{20}) >$$

where:

- $M_A(Mo_1, Mo_{10}, Mo_{20}) = Avg(f_1(Mo_1, Mo_{10}) f_2(Mo_1, Mo_{20}), 0.5, 0.8) > 0.7$.
- $Mo_1 = predBob.jpg$.
- Mo_{10} and Mo_{20} represent the multimedia objects.

The template T_1 is satisfied only if its M_A is satisfied (which means that the score returned from the average aggregation function is greater than 0.7).

6.5 Uncertainty Resolver *via* Aggregation Functions

Integrating multimedia data and functions (face recognition, image similarity, object recognition, etc.) in Ambient intelligence environments may lead to complex situations and thus *uncertain* decisions to take. As mentioned before, this comes from several assets (lightings, electrical noise, functions' relevance, etc.) and affects the related result. In order to validate a given fact and to avoid error risk, one should consider relying on several sources (e.g. taking several snapshots of an environment) using various multimedia functions to retrieve the most appropriate result for a given situation. This is why, we introduce here the concept of *aggregation functions* aiming to reduce uncertainty by filtering andor aggregating a set of values in order to select or compute one relevant value for facilitating decision-making. An *aggregation function* μ can be illustrated as in Figure 6.1 and defined by any probabilistic function such as the combination rule of Dempster and Shafer theory of evidence (DS) [27] [11] Bayesian Decision theory [10], Decision Trees [9], the average, the minimum, the maximum, and so on. It is formally written as:

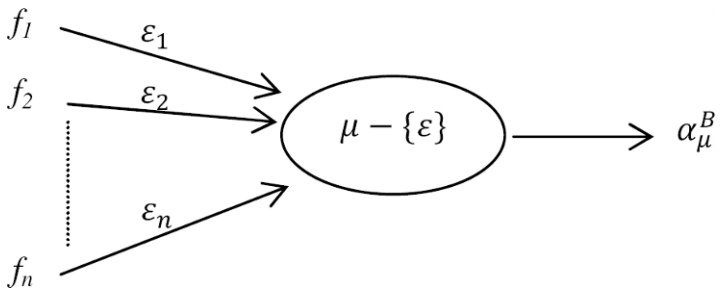


Fig. 6.1 An aggregation function representation.

$$\mu(\{f_1(MO_k, MO_l), \ldots, f_n(MO_k, MO_l)\}, \{\varepsilon_1, \ldots, \varepsilon_n\}) \rightarrow \alpha_\mu^B \tag{6.5}$$

where:

- f is a multimedia function,
- εan uncertainty threshold ($\in [0,1]$) representing the percentage of noise that can affect the result. In fact, ε can affect the overall thresholds returned by the multimedia functions or it can be related to each of the thresholds. In that case, the noise (ε) can be automatically calculated based on the difference between the environment state of the predefined *MO* and the state when the instances are captured (i.e. lighting changes and background detection). If omitted, $\varepsilon = 0$ meaning that no uncertainty detected.
- α_μ^B is the filtered confidence score ($\in [0,1]$) of a Boolean value B with respect to the aggregation function.

Aggregation process defined here handles multimedia functions with probabilistic values that depend on the classification and relevance of the response they uncover. However, functions with different treatment cannot be invoked in the aggregation process or they should be normalized to suit the corresponding format. In the following, we present and detail how we adapted several existing aggregation functions such as *average-based*, *Bayesian Network-based*, *Dempster and Shafer-based*, and *decision trees-based* functions.

6.5.1 Average-based Function

One of the most used aggregation function is average function. However, it cannot be used as it is in our approach as we need to integrate the uncertainty threshold related to each multimedia function (or the overall uncertainty threshold). The proposed adapted average function to filter out the set of results returned by the multimedia functions including the uncertainty thresholds is defined as follows:

$$Avg\left(\{f_1(MO_k,MO_l),\ldots,f_n(MO_kMO_l)\},\{\varepsilon_1,\ldots,\varepsilon_n\}\right)=\frac{\sum_{i=1}^{n}\alpha_i^B}{n+\sum_{i=1}^{n}\varepsilon_i}\rightarrow\alpha_{Avg}^B$$

where:

- α_i^B represents the thresholds returned by the multimedia functions
- ε_i is the uncertainty threshold for a given multimedia function result. It is important to note that $\sum_{i=1}^{n}\varepsilon_i$ can be replaced by ε if an overall uncertainty threshold is defined in the aggregation function.

Let us take the multimedia functions defined earlier f_1 and f_2 which return for each instance image seized a related result (e.g. $f_1\rightarrow 0.5^T$ and $f_2\rightarrow 0.8^T$). These thresholds are considered as 2 different results for the same fact (detecting the presence of *Bob*). Let $\varepsilon=0.2$ be the predefined overall uncertainty threshold. After applying the *Avg* function, the computed result becomes:

$$Avg\left((0.5^T,0.8^T),0.2\right)\rightarrow\frac{0.5^T+0.8^T}{2+0.2}=0.59^T$$

The decision is then made by comparing the predefined threshold with the calculated one.

6.5.2 Bayesian Network-Based Function

Bayesian network (BN) is a probabilistic graphical model used to represent a set of variables and their probabilistic independencies. The graph G is defined as G= (V, E) where V is a set of vertices representing the variable of interest and E the dependencies relationship between these variables. Each random variable Vi can hold a finite set of mutually exclusive states and has a conditional probability table $p(V_i|\pi(V_i))$

where $\pi(Vi)$ represents the parent set of V_i. The Bayesian Network encodes a joint probability over the set of variables defined by the following chain of rule:

$$P(V) = \Pi_{i=1}^{n} P(V_i|\pi(V_i))$$

Hence, the BN is defined by the structure of the graph and by the conditional probability table of each variable. BNs have been applied to different domains including several intrusion detection techniques [28] [29] and Knowledge based authentication [30]. Due to its reliability, we chose to adapt in our approach a naive BN in order to aggregate multimedia functions returned results. The idea is based on the assumption that the validation of a given concept (such as detecting the presence of *Bob* in a given input image) takes place when several multimedia functions (or several input snapshots) returning a set of thresholds are filtered and their returned threshold is computed against a predefined threshold. Formally, we define the following notations and terms:

- C denotes a class variable related to the concept claimed to be valid or not. The possible outcomes of C are defined as True or False depending on its state.
- We define $f = f_1,, f_i$ representing the selected subset of variables related to the concept C. Each of the variables has the parent C and represents a factoid with a possible binary outcome {True, False} indicating whether the concept C is valid or not. The set of multimedia functions f (or one function with several input images to process) are considered variables related to C where the similarity values they compute affect the result of C.

Given the joint distribution, the probability of the class C for a True Threshold value can be calculated using the Bayes' rule:

$$P(C = T|e) = \frac{P(e|C = T)P(C = T)}{P(e)} = \frac{\Pi_{e \subseteq f} P(e|C = T)P(C = T)}{\sum_{b=T}^{F} \Pi_{e \subseteq f} P(e|C = b)P(C = b)}$$

$e \in f$ and denotes the set of variable of interests representing different multimedia functions. The returned filtered result is equal to the value of $P(C = T|e)$. As a result, the BN is described as follows:

$$BN(\{f_1(MO_k, MO_l), \ldots, f_n(MO_k, MO_l)\}, \{\varepsilon_1, \ldots, \varepsilon_n\}) \rightarrow P(C = T|e)$$

To estimate the values of the conditional probability distributions for each node of the graph, we refer to the results returned by each of the multimedia functions designating the variables of interests. To preserve accuracy, we consider the probability of the parent Class C for a given value (true or false) equal to 0.5 which means that there exists 50% that the concept represented by the class C is valid and 50% otherwise. Referring to our motivating scenario, the template *Face_Identification* contains the M_A Boolean expression with two multimedia functions f_1 and f_2. The parent class C represents the concept of detecting the presence of *Bob* in the captured images. Let us assume now that f_1 returns a confidence score of 80% and f_2 a confidence score of 60%. As mentioned before, the probability of the given concept

is equal to 0.5 for both binary values. However, the conditional probability distribution $P(f_1|C)$ provided in Figure 6.2 is described as follows. When the concept is valid, then $P(f_1 = T) = 0.8$ which is the returned multimedia function's result, and $P(f_1 = T) = 0.2$ otherwise. The same computation is applied on $P(f_2|C)$.

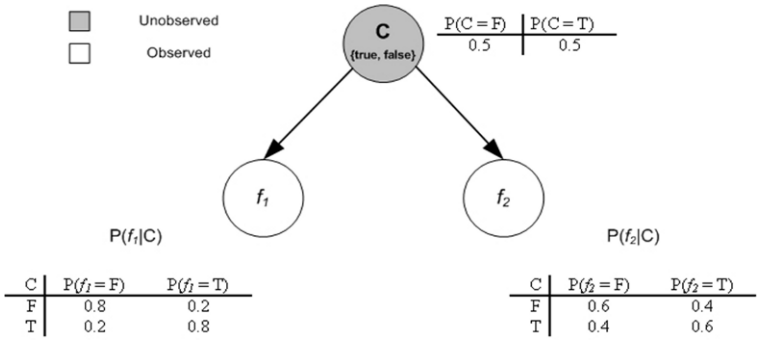


Fig. 6.2 A BN applied example.

Finally the computed result is given by:

$$\begin{aligned} &BN(\{f_1(Mo_1, Mo_{l0}), \ldots, f_n(Mo_2, Mo_{20})\}, \{0,0\}) \\ &= \frac{P(f_1 = T, f_2 = T|C = T)P(C = T)}{P(e)} \\ &= \frac{\Pi P(f_1 = T, f_2 = T|C = T)P(C = T)}{\sum_{b=T}^{F} \Pi P(f_1 = T, f_2 = T|C = b)P(C = b)} = 0.85 \end{aligned} \tag{6.6}$$

However, the BN function described above calculates a resulting value without taking into consideration the uncertainty threshold specified for each multimedia function. Thus, integrating uncertainty threshold decreases the possibility of unauthorized access. In the BN function, is deduced from the observed nodes and target their observed values (see Figure 6.3).

Now, let us consider the same motivating example with an uncertainty threshold equal to 0.1. After applying Bayes' rule, the BN function computes the following value:

$$\begin{aligned} &BN(\{f_1(Mo_1, Mo_{l0}), \ldots, f_n(Mo_2, Mo_{20})\}, \{0.1, 0.1\}) \\ &= \frac{\Pi P(f_1 = T, f_2 = T|C = T)P(C = T)}{\sum_{b=T}^{F} \Pi P(f_1 = T, f_2 = T|C = b)P(C = b)} = 0.75 \end{aligned} \tag{6.7}$$

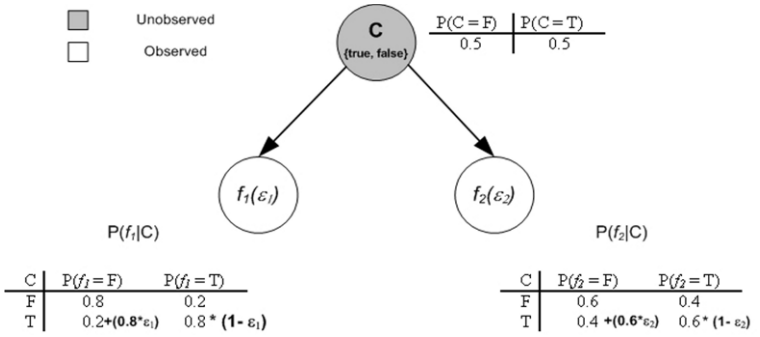


Fig. 6.3 A BN with deduced uncertainty threshold.

6.5.3 "Dempster and Shafer"-Based Function

Dempster and Shafer function is based on the mathematical theory of Dempster and Shafer which is used to calculate the probability of an event given a set of evidences. In this section, we provide an overview of the function with its possible adaptation to fit our objectives.

1. *Frame of discernment* (τ): represents the set of elements in which we are interested. In our approach, $\tau = True, False$ where the values *true* or *false* represents a result for a given fact (e.g. *Bob* is detected). Given the elements in τ, 2^{τ} denotes all possible propositions that could describe τ represented as: $P(\tau) = \phi, True, False, True, False$.
2. *Mass function* (m): can be compared with a degree of confidence of an element. It is a basic probability assignment belonging to [0,1] which defines a mapping of the power set where 1 stands for a total confidence and 0 for no confidence at all.
3. *Dempster's rule combination*: is used for gathering information to meaningfully summarize and simplify a corpus of data whether the data is coming from a single source or multiple sources. In our case, input multimedia objects can be acquired from one source or several sources. For this reason, each result returned from a multimedia function and related to an input multimedia object is considered as evidence with a calculated confidence. For instance, consider that the multimedia functions f1 and f2 would return for each instance image seized a related result (e.g. $f_1 \rightarrow 0.5^T$ and $f2 \rightarrow 0.8^T$) which are considered two different results for the same fact (detecting the presence of *Bob*). Given the two different evidences (representing the different thresholds calculated for the same fact) to support a certain proposition A, the rule combines them into one body of evidence. Thus, the rule determines a measure of agreement between the 2 evidences using:

$$m_{12}(A) = m_1 \otimes m_2(A) = \frac{\sum_{B \cap C = A} m_1(B) m_2(C)}{1 - \sum_{B \cap C = \phi} m_1(B) m_2(C)} \; when \; A \neq \phi \tag{6.8}$$

In our approach, we use the combination rule specified above to aggregate the returned result of a multimedia functions in order to get one representing threshold. In conjunction with the returned threshold, the multimedia predicate is evaluated.

Let us go back again to our motivating scenario in which we wish to detect the presence *Bob*. For this reason, the system uses a *face_identification* template with a M_A holding

- The combination rule of Dempsfer and Shafer's theory of evidence (DS) as aggregation function with an uncertainty threshold equal to 0.1
- The two multimedia functions f_1 and f_2 described above are used to analyze multimedia objects.
- The predefined Multimedia objects Mo_1, Mo_2 representing *Bob*.

Let us assume now that f_1 returns a confidence score of 80% according to the captured snapshot Mo_{10} and f_2 a confidence score of 60% according to the captured snapshot Mo_{20}. These scores represent the probability of validating the concept of detecting the *Bob* in the snapshot images. They are related to the mass functions (m_i) defined in the DS theory of evidence. For instance, in our case $m_1(true)$ holds the probability of detecting the *Bob* which is determined using the first function whereas $m_2(true)$ holds the same concept thus determined with the second function. According to the DS theory combination rule, we can compute the aggregated confidence score as follows:

$$m_{12}(true) = \frac{m_1(true) \times m_2(true) + m_1(true) \times m_2(true, false) + m_1(true, false) \times m_2(true)}{1 - K} \tag{6.9}$$

where $K = m_1(true) \times m_2(false) + m_1(false) \times m_2(true)$ represents the conflict in the combination rule

In order to reflect the uncertainty when dealing with multimedia objects, the uncertainty threshold predefined for the combination of the multimedia functions f_1 and f_2 in the aggregation function is deduced from the probabilities of the concept to be determined:

- $m_1(true) = m_1(true) - m_1(true) \times \varepsilon = 0.8 - 0.8 \times 0.1 = 0.72$
- $m_2(true) = m_2(true) - m_2(true) \times \varepsilon = 0.6 - 0.6 \times 0.1 = 0.54$

However, if m12(false) is being calculated, (the fact that the Bob is not detected in the captured image), the threshold would be deduced from m1(false) and m2(false)

- $m_1(true, false) = 1 - (m_1(true) + m_1(false)) = 0.08$
- $m_2(true, false) = 1 - (m_2(true) + m_2(false)) = 0.06$

After applying the combination rule, we obtain the following aggregated confidence value:

$$m_{12}(true) = 0.786$$

which, with respect to the previous parameters and assumptions, conducts the application to consider that *Bob* is not identified within the provided snapshot as it is below the acceptance threshold defined in the predicate.

6.5.4 Decision Tree-Based Function

Decision Trees (DT) are one of the supervised machine learning techniques based on logic methods of inferring classification rules. The most well-known decision tree induction algorithms for statistical uncertainty are ID3 [9] and its successor C4.5 [25]. Decision trees provide a solution applicable on situations that cover cognitive uncertainty, in particular vagueness and ambiguity. In the following, we show how we adapted the approach proposed by Y. Yuan and M. Shaw in [31] considered as predecessor of many other works related to decision trees with fuzzy concept. To help explaining this adaptation, we will illustrate each definition with the use of our motivating example. The input of the DT function is a case u belonging to a universe U (where $U = u$). In our example, we have two cases: u_1 is the first input where (True = 0.8, False = 0.2) and u_2 is the second input where (True=0.6 and False=0.4). Each u_i is described by a set of attributes $A = A_1, \ldots, A_k$ In our example, we have only one attribute which is the Source of providing values (let's say that a source can be simply a multimedia function). A_i has various linguistic terms $T = T_1, \ldots, T_k$ where $T(A_j)$ is a possible value for the attribute A_j. Here, the T has two values True and False, with a certain percentage for each one. Finally, each case will be classified in a class $C = C_1, \ldots, C_j$ which is the final result of the DT function, also called decision attribute. So:

$$\begin{aligned} A &= Source1 \\ TSource1 &= True, False \\ C &= True, False \end{aligned} \tag{6.10}$$

We will also use the function ρ to represent the membership degree of an object to an attribute. It returns any value in the interval $[0,1]$. In our example, ρ will be the values associated to terms (true, false) by the source (e.g. true=0.6, false=0.4).

	Source		**C**	
U\T	True	False	True	False
u_1	0.8	0.2	0.8	0.2
u_2	0.6	0.4	0.6	0.4

Table 6.1 Sample Data Set.

Considering the sample data set provided in Table 6.1 where n input values from a source are provided to represent how much Bob (the person to be authenticated) has been detected in the captured images using a multimedia function. With the first input, Bob has been 80% detected, and with the second one he has been 60% detected. The values of the column C are the average of all the T terms of all the attributes A. In this scenario, we have only one attribute "Source", this is why the

True value of the decision attribute C is the same as the true value of the attribute Source. If we had two sources, Source1 and Source2 for example, with a value of 80% for True in Source1 and 60% for True in Source2, the True value of the decision attribute would be the average of the both sources which is 70%. Drawing the DT is not important in our adaptation. We only need to compute the ambiguity after each input. For the lowest ambiguity reached, we retrieve its values for the true and false attribute and use them for the final decision. To do so, we need to define the classification ambiguity G(P), with fuzzy portioning P. It measures the classification and is computed as the weighted average of classification ambiguity with each subset of the partition as follows:

$$G(P) = \sum_{i=1}^{k} w(E_i) \times G(E_i)$$

where:

- $G(E_i)$ is the classification ambiguity with fuzzy Evidence E_i,
- $w(E_i)$ is the weight that represents the relative size of subset E_i as:

$$w(E_i) = \frac{M(E_i)}{\sum_{j=1}^{k} M(E_j)}$$

  where M is the cardinality measure (or sigma count) of a fuzzy set A, defined by $M(A) = \sum_{u \in U} \mu_A(u)$ which is the measure of the size of A.

For instance, G(Source) of our sample data set provided in Table 1 will be computed as follows:

- After the first input:

$$\begin{aligned} G(source) &= w(True) \times G(True) + w(False) \times G(False) \\ &= 0.8 \times 0.17 + 0.2 \times 0.69 = 0.27 \end{aligned}$$

- After the second input:

$$\begin{aligned} G(source) &= w(True) \times G(True) + w(False) \times G(False) \\ &= 0.7 \times 0.29 + 0.3 \times 0.69 = 0.41 \end{aligned}$$

As we can see, after the second input, the ambiguity value is higher than the ambiguity value after the first input, thus we select the values of the lowest ambiguity and consider as a final result a detection confidence score of 80%. The adaptation of fuzzy decision tree to our approach allows us to select the input that engenders the least ambiguity. This can be extended to deal with many inputs coming from many sources; the most trusted source with the best engendered input is selected by retrieving the least ambiguous source and the least ambiguous input value as well.

6.6 Experimentation

In this section, we present a set of experiments elaborated to study the impact of using different aggregation functions and show how uncertainty can be reduced. The experiments were conducted using one PC Intel Pentium M having 1.73 GHz processor speed and 1GB RAM. We plugged a WebCam with 1.3 Megapixel resolution in video mode and 5 Megapixels in image mode to capture real time snapshots. The experiments were made in one of our laboratory rooms with various lighting conditions. Thus, we defined three profile environments:

- P_1: representing a maximum lighted environment,
- P_2: representing normal lighted environment
- P_3: representing minimum lighted environment.

The objects used to conduct the set of experiments are human faces, random objects and laboratory rooms to represent locations. We used the same multimedia functions described in our motivating scenario in Section 6.1:

- f_1: is related to the InterMedia Oracle module, used for location and object identification. Predefined images needed for f_1 are stored in an Oracle 10g database.
- f_2: is an SVM classifier based on color object recognition and used for face identification and object identification.

Whereas, the set of aggregation functions we used are:

- DS: representing the Dempster and Shafer - based function
- BN: representing the Bayesian Network - based function
- DT: representing the Decision Tree- based function
- Avg: representing the Average-based function
- Min and Max functions referring to the minimum or maximum value returned from the set of results of the multimedia functions.

The conducted experiments are divided into 3 different steps described as follows:

1. Aggregation Function Accuracy: to calculate the accuracy and time processing of each of the aggregation functions used
2. Value Distribution: to show the evolution of the aggregation functions results according to a set of manually generated values with different distributions
3. Template Tuning: refers to finding the appropriate template according to variable environmental conditions. In this test, we process the captured images under different profiles (P_1, P_2, P_3) using several templates in order to determine the appropriate one in each profile.

In the following, the results returned from the templates refer to the results of the multimedia attribute expression defined before its comparison with the predefined threshold. In other terms, they represent the results returned from the aggregation function $\mu(f_1(MO_k, MO_l), \ldots, f_n(MO_k, MO_l), \varepsilon_1, \ldots, \varepsilon_n) \rightarrow \alpha_\mu^B$

6.6.1 Aggregation Function Accuracy and Time Processing

When processing multimedia objects in Ambient Intelligence environments, unpredictable factors related to the context and the human behavior could affect the processing result and lead to positive or negative decisions considered frustrating in both cases. With the use of the uncertainty resolver and some aggregation function(s), we intent to minimize the probability of making faulty decisions by relying on several sources or captured snapshots w.r.t.[4] to the processing time. Now, once we get the set of scores returned from the multimedia functions, the system should interpret and aggregate adequately these sets of scores into one and unique relevant score. The aim of this test is to study the effect of integrating aggregation functions to reduce faulty decisions. We used the multimedia function f_1 and a set of learned images representing different objects which one of these objects is the face of Bob (the person to detect). In the first part of the test, we repeated 10 times the fact of capturing one snapshot of Bob (without invoking the aggregation function) whereas in the second part of the test, we repeated 10 times the fact of capturing 5 snapshots of Bob with different behaviors, and filtered them using the different aggregation functions. The accuracy of the aggregation functions is determined by comparing the aggregated results returned after 5 snapshots of Bob and the results of one snapshot of Bob without invoking the aggregation functions. The obtained results are shown in Table 6.2 and Table 6.3.

Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9	Test10
0.702	0.364	0.609	0.514	0.609	0.562	0.603	0.512	0.484	0.517

Table 6.2 Values returned by multimedia functions without aggregation.

	Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9	Test10	Accuracy
Min	0.601	0.649	0.596	0.599	0.618	0.566	0.496	0.554	0.994	0.663	70%
Max	0.748	0.957	0.876	0.882	0.918	0.680	0.575	0.686	0.64	0.704	90%
Avg	0.676	0.974	0.923	0.927	0.950	0.648	0.541	0.615	0.731	0.692	80%
DS	0.986	0.704	0.607	0.620	0.629	0.665	0.575	0.686	0.795	0.744	80%
BN	0.976	0.704	0.633	0.637	0.635	0.956	0.698	0.915	0.996	0.693	100%
DT	0.748	0.602	0.553	0.556	0.585	0.665	0.575	0.686	0.795	0.744	70%

Table 6.3 Values filtered after 5 captured snapshots for each test.

According to the tables above and considering the fact that Bob appeared in several captured snapshots, we calculate the accuracy for each aggregation function i as follows:

[4] With respect to.

$$Accuracy(i) = \frac{\sum_{j=1}^{n} P_j(Table_2(j) < Table_3(i,j))}{n} \tag{6.11}$$

where

- $Table_2(j)$ is the content of the j^{th} column, and $Table_3(i,j)$ is the content of the i^{th} line and j^{th} column
- $$P_i(Table_2(j) < Table_3(i,j)) = \begin{cases} 1 \; if \; Table_2(j) < Table_3(i,j) \\ 0 \; if \; Table_2(j) > Table_3(i,j) \end{cases} \tag{6.12}$$
- n is the number of tests elaborated.

All of the aggregation functions returned an accuracy score relatively high which obviously prove their ability to minimize the risks of having false decisions when being integrated in a decision making system. In addition, Table 6.3 shows that the BN provides a maximum accuracy value. This means that according to the set of tests elaborated here, it is the most accurate aggregation function. Of course, this can change in other contexts.

In addition to the accuracy tests, we studied the time processing of each aggregation function when integrated in a decision making system. The objective of this test is to show the influence of aggregation functions on the overall system performance. The set of input images were processed using multimedia function f_2. Figure 6.4 shows the results (in ms) of an incremented set of images starting from 2 inputs to 11 inputs tested on each aggregation function.

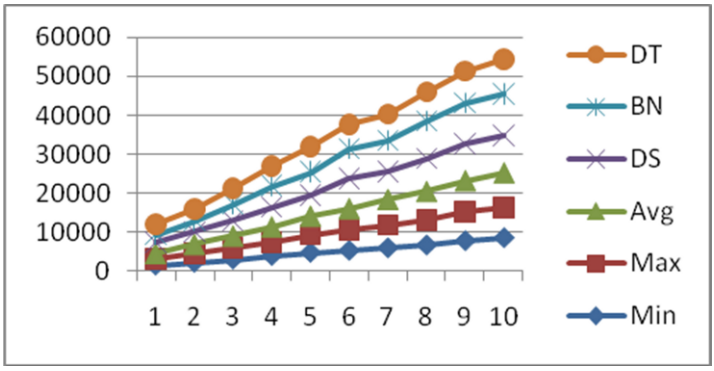


Fig. 6.4 Processing Time Results.

As we can see in here, the overall results are linear reflecting the fact that time increments with the number of inputs. The *Min* and *Max* functions need the minimum time to perform its aggregation while DT aggregation function requires the maximum time (we think that it is related to an implementation/optimization issue that we will solve soon). To conclude the first step of our tests, we can say that the accuracy and time processing of aggregation functions of our proposed approach sound obviously practical.

6.6.2 Value Distribution

In this experimental step, we elaborated a set of tests to observe aggregation functions attitude according to randomly generated values between 0 and 1. These set of values represent the degree of certainty in percentage regarding the recognition and identification of an object, person or location. We detail each set below.

6.6.2.1 Test 1: Values higher than 0.5

In this test, a set of 200 random values of a range between 0.5 and 1.00 was generated. As we will see, two different investigations of the 200 generated values were carried out each time with a different average value in order to evaluate aggregation functions while varying the Uncertainty Threshold (UT).

In the first investigation, the average of the values was fixed to 0.95 (Figure 6.5). Here, we notice that the *Min* function is decreasing linearly as the UT is increasing. The highest value for this function is reached at UT = 0 and around 0.5 less than all the values of the other functions. The Max and the Avg functions decrease moderately while having close values. The DS and the BN functions remain constant until the BN reaches UT = 0.4 and DS reaches UT = 0.9 when they collapse dramatically to 0. The same result (=1) is for all UT, meanwhile all the others show a linear decrease. The DT function starts with values close to the Max function and goes down moderately until reaching UT = 0.5 where a quick drop of its values makes them move toward the value of the *Min* function.

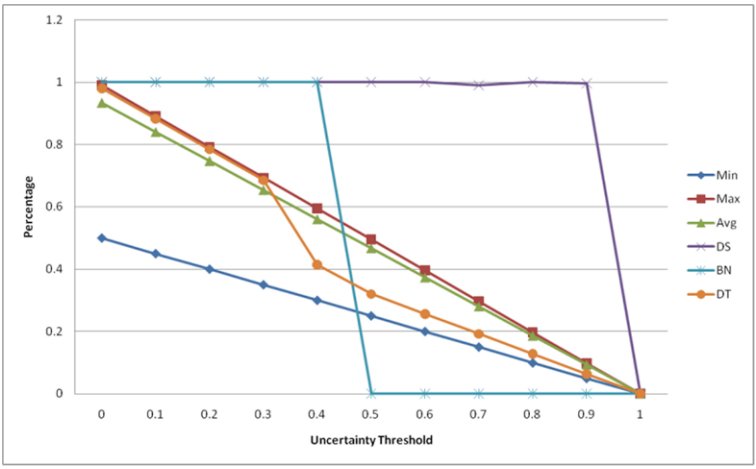


Fig. 6.5 Evaluating aggregation functions with an average value = 0.95.

In the second investigation, the average of the values was decreased to 0.75 (Figure 6.6). The *Min* as well as the *Max* functions keep the same decline as in the first

investigation, meanwhile the Avg shows the same decline as UT increases. However, we notice here that the overall average of the values is less than the ones in previous investigation. The BN converges to 0 at 0.4 (while it converged to 0 in the previous one at UT = 0.5). The same observation for the DS function that converges on UT = 0.7 in this investigation (and on UT = 0.9 in the previous one). Also, the DT function starts with values close to the *Max* function and it quickly drops down at UT = 0.3 and its values become closer to the *Min* function after UT = 0.4.

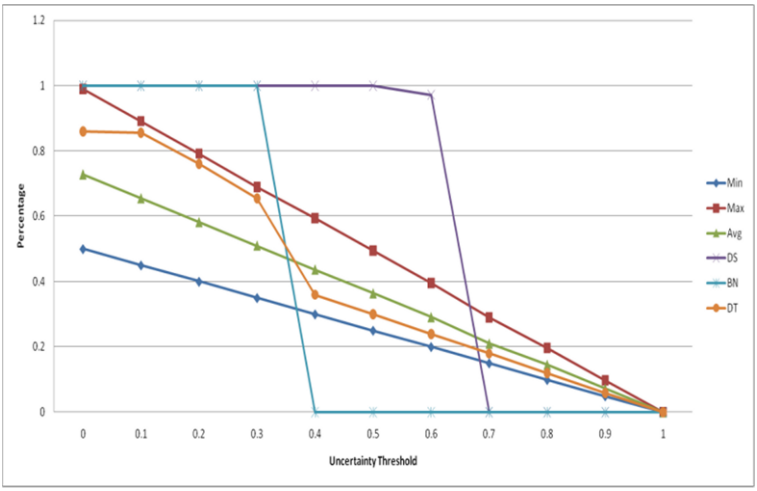


Fig. 6.6 Evaluating aggregation functions with an average value = 0.75.

6.6.2.2 Test 2: Values less than 0.5

In this test, a set of 200 random values of a range between 0 and 0.5 was generated. As we can see in Figure 6.7, we can observe that the *Min*, DS and BN functions are just a bit over 0. For the three other functions, the values regularly plunge to reach the lowest level 0 at UT = 1. Concerning the DT function, its values are close to the *Avg* function values. *Max* function has the highest values here.

6.6.2.3 Test 3: Random Values

In this test, random values with no restrictions were generated and used as inputs of the aggregation functions. Two generated sets were used (Figure 6.8 and Figure 6.9). In the first set (Figure 6.8), the most obvious observation is that the DT function started with a value that increases starting from UT = 0 to reach its maximum of UT = 0.2 and then it decreases regularly.

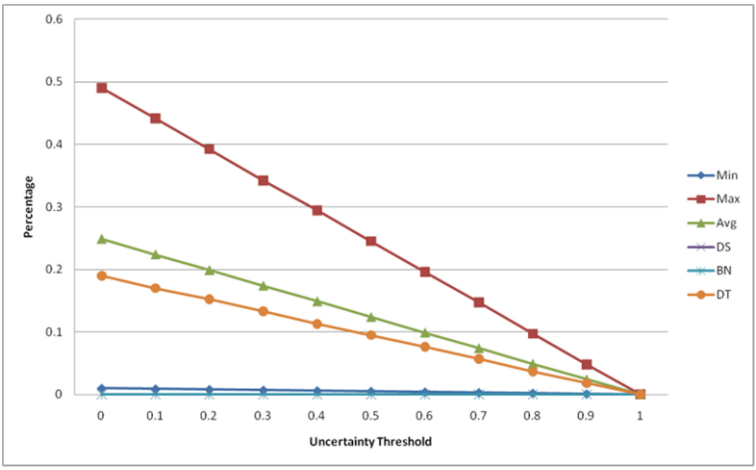


Fig. 6.7 Evaluating aggregation functions with an average value = 0.25.

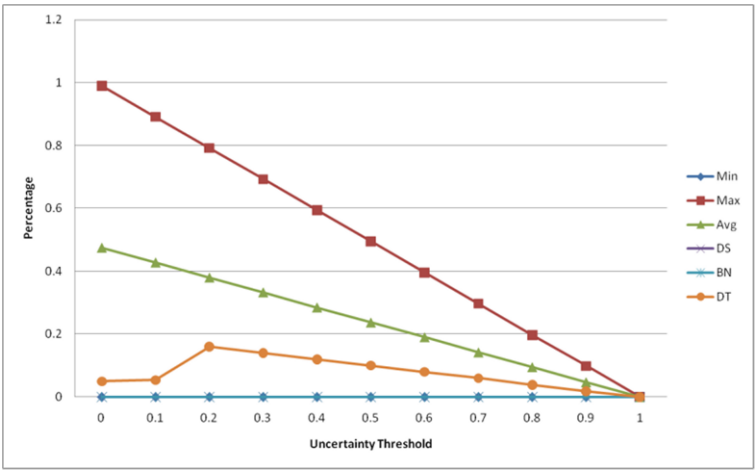


Fig. 6.8 Evaluating aggregation functions using low average random values set.

In the second set having higher average random values (Figure 6.9), except the BN function, all the others were not affected by this change.

The result of this test is shown in Figure 6.10. Here, the BN function remains constant with a value of 1 for the entire test. The *Min* function returns, as always, the lowest values. The DS function converges to 0 with a sudden change from 1 to 0 on UT = 0.3.

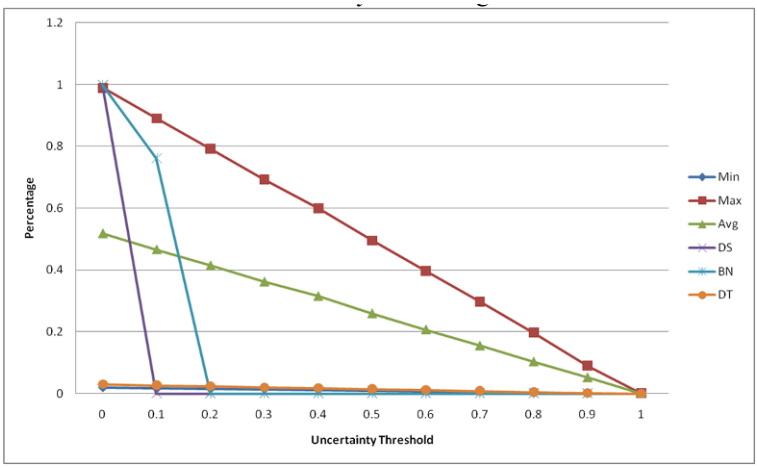


Fig. 6.9 Evaluating aggregation functions using higher average random values set.

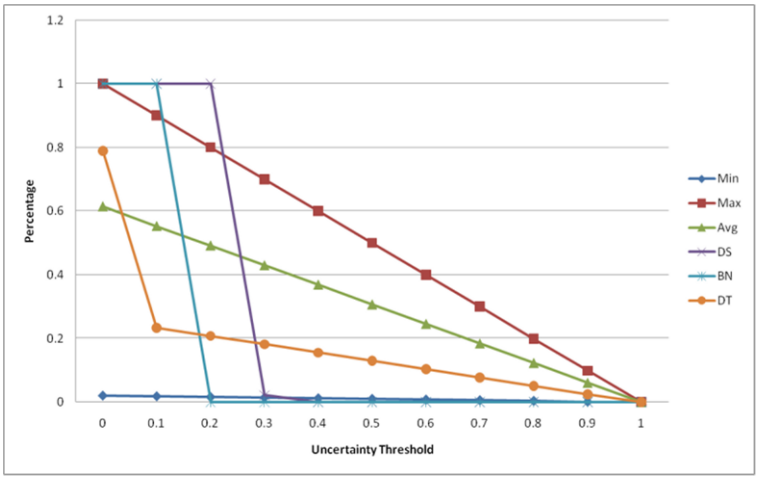


Fig. 6.10 Evaluating aggregation functions where 75% of the values are higher than 0.5 with an average of 0.62.

6.6.2.4 Test 5: 75% of the values are less than 0.5

In this test (Figure 6.11), once the majority of the values becomes closer to 0, the DS, BN, and *Min* functions return 0. The DT shows here an important decline when uncertainty threshold is located between 0.1 and 0.2.

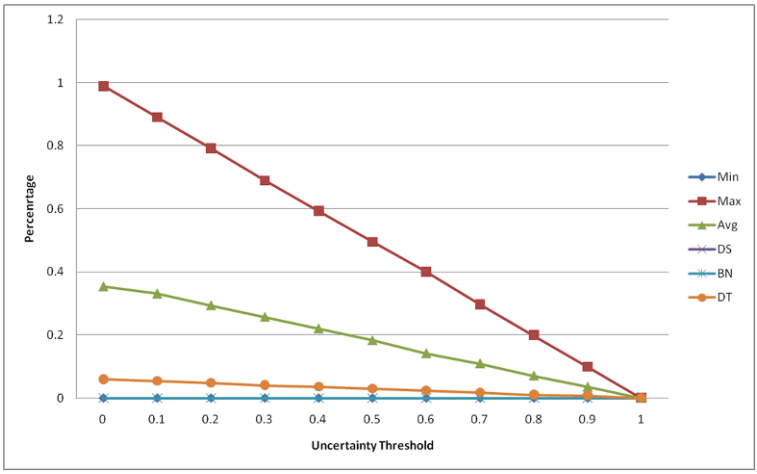


Fig. 6.11 Evaluating aggregation functions where 75% of the values are lower than 0.5 with an average of 0.35.

6.6.2.5 Test 6: Equally distributed values

Here again in this test, the *Min*, DS, and BN functions have 0 for the entire test, meanwhile the DT shows for the second time a small raise followed by a slow decline.

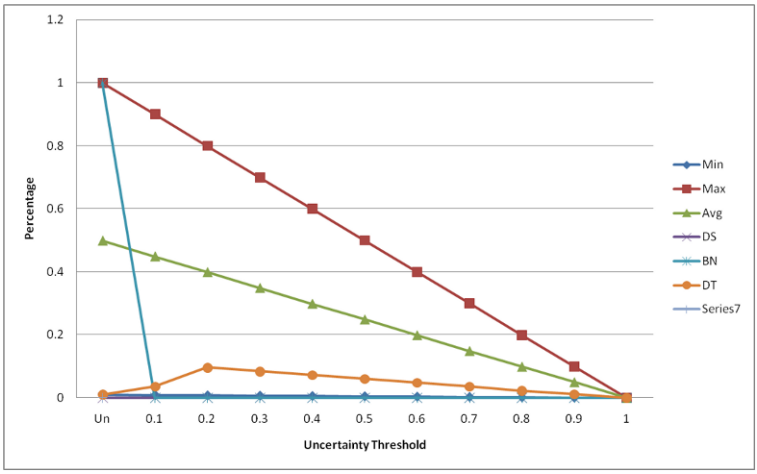


Fig. 6.12 Evaluating aggregation functions with equally distributed values (50% lower and 50% higher than 0.5).

6.6.2.6 Test 7: Distribution change

In this test, a set of 200 values were generated. Initially, 0% of the values were less than 0.1. After, with each iteration, we increased the percentage continuously to study the behavior of the functions on each percentage (Figure 6.13). Since the maximum values were not a variable key in this test, the *Max* function is constant. The *Min* function has the following behavior: it has initially 0% of the values less than 0.1% before it promptly drops down when 10% of the values become less than 0.1. The *Avg* function is decreasing slowly with each distribution. The DT behaves similarly than the *Min* function when dropping down but it conserves slightly higher values along the entire test. The DS function provides a sudden change when 25% (precisely between 25% and 26%) of the values become less than 0.1; the same also happens with the BN after 25% of the distribution become less than 0.1 before reaching 0 when 30% of the distribution values are less than 0.1.

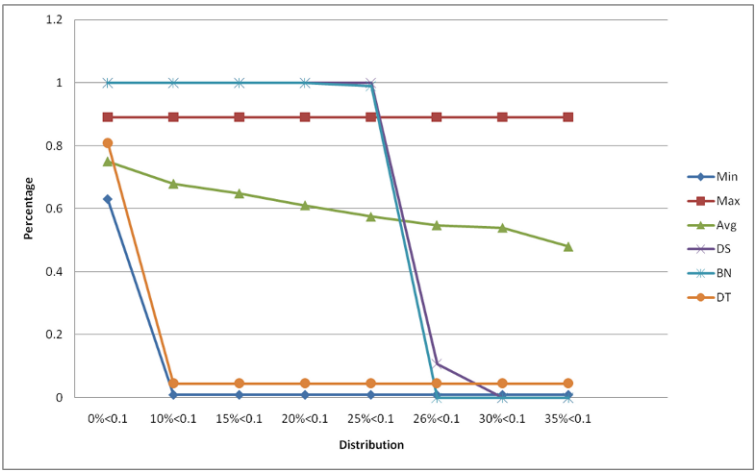


Fig. 6.13 Aggregation function behavior when distribution values change.

6.6.2.7 Test 8: Influence of the number of returned values 0 and 1 on the aggregated result

The aim of this test is to study the impact of the number of returned values 0 and/or 1 on the decision. Figure 6.14 shows the computation results and aggregation function behavior that we try to explain here:

- *Min* and *Max* functions: the *Min* function returns 0 when at least only one ”0” exists in the distribution. Similarly, the *Max* function returns 1 when at least only one ”1” value appears in the distribution.

- *Avg* function: the presence of 0 and 1 in the value distribution has not much influence on the *avg* function.
- DS, BN, and DT functions: here, several cases can be identified:
  - " if 99% of the values in the distribution are 1 and 1% contains 0 values, then DS, BN and DT functions output 0 as a result. If this 1% contains any number ($\neq 0$), the output is always 1.
  - " if 99% of the values are higher than 0.5 and only one value is 0 then:
    - DS and BN functions return 0.
    - DT function returns 0 when the ambiguity of a successive set of values gets lower than the current lowest ambiguity.
  - if 99% of the values are less than 0.5:
    - when 33% of the values are 1, DS function starts returning results higher than 0 and then it starts returning values close to 1 at 36%,
    - when 36% of the values are 1, BN function returns positive results and quickly after returns values that are equal 1 starting a percentage of 39%
    - DT function returns 1 when the ambiguity of a successive set of values gets lower than the current lowest ambiguity

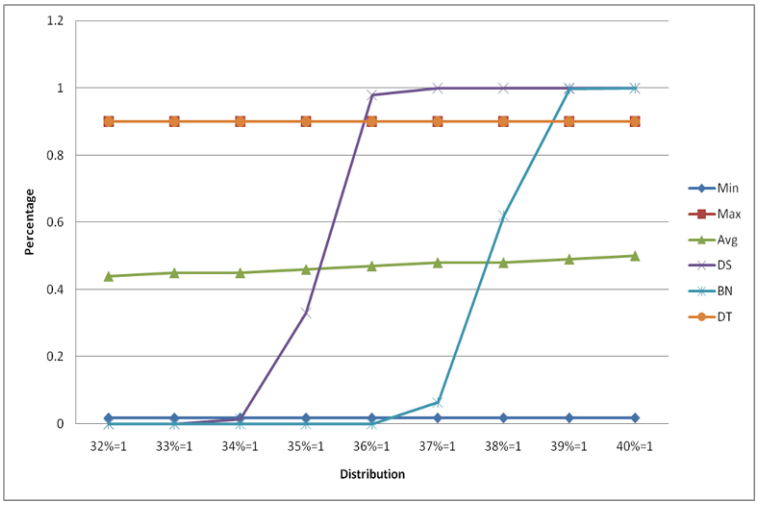


Fig. 6.14 Influence of the percentage of the 1 value in a distribution.

6.6.2.8 Discussion

Through this step of our experimentations, we aimed at studying the evolution and the behavior of the aggregation functions in response to different value distributions

and threshold variations. Based on the computed results, we are able to pin down several observations.

The *Min* function returns the minimum values among all the experiments. This means that it ignores the sensitivity of the variations of values (we say it is variation insensitive) and can be very useful when maximum security is required in an application.

Similarly, the *Max* function returns the highest value in the distribution which can be used in several applications to alert and/or prevent users.

The DT function has a similar behavior with the *Min* function when uncertainty threshold starts to increase ($UT >= 0.5$) and particularly when values are less than 0.5. However, in the distributions where the values are neither too high nor too low, the DT function returns nearby average values. This makes the DT function more practical to be used in applications where security is moderated. In addition, an important observation has been seen in Test 3 about the DT function where it starts with a value that increases starting from UT = 0 to reach its maximum with UT = 0.2, and then decreases continuously. This shows that when the uncertainty level increases, the returned results cannot decrease but increase as well. This is the law of uncertainty.

We also concluded that the BN function is the most sensitive to variations. The DS function is similar but less sensitive. This has been observed with tests having small value changes where the DS and BN had a sudden change. When we proceeded with test 7, we had initially 0% of the value less than 0.1 and 100% of the values higher than 0.6. Then, we started exchanging the high values with values less than 0.1. When the percentage of the values less than 0.1 becomes 25%, the DS function converges to 0, meanwhile the BN needs 30% to converges to 0. This means that these two functions can be used in high secured applications having very sensitive cameras with powerful hardware.

In test 8, as mentioned earlier, we can mainly observe that for the DS and the BN, a single 0 value in the distribution makes the output result 0 even if all the other values are 1. On the other hand, if all the values are 0 and we replace each 0 with a value of 1, the DS needs 38% of the values to become 1 so that the result increases towards 1, meanwhile the BN needs 40% to start increasing.

6.6.3 Template Tuning

The objective of this step consists of determining the appropriate template for each profile representing a given environment. This would help the system administrator to tune his platform regarding the different environments that could affect the results of multimedia objects processing in an Ambient Intelligence environment. We proceed with the pre-analysis phases needed to start the experiments: the learning and template generation phases that we describe below:

1. Learning phase: the system learns to identify a concept which represents a human face, an object or a location. It refers to determining the set of predefined

multimedia objects in the different profiles. We specify 3 different set of predefined images taken in the 3 given profiles: B_1, B_2, and B_3 representing a set of predefined images of a given object captured in P_1, P_2, and P_3 respectively.
2. Template generation: here, we defined 6 different templates (T_1, T_2, T_3, T_4, T_5, T_6) associated to the aggregation functions Min, Max, Avg, DS, BN and DT respectively.

Thus, multimedia functions are used alternatively depending on the object to be detected. In case of a location or object identification, we used the f_2 multimedia function, whereas for face identification, we used the f_1 multimedia function. We proceeded as follows: a number of images are captured using our webcam and processed according to the defined templates holding multimedia and aggregation functions. The multimedia functions compare the captured images with the set of predefined images stored for each object, and return a set of similarity scores which are aggregated later on according to the aggregation function associated to the related template. And so, the appropriate template is determined based on the highest returned value in each profile (assuming that the captured images of the object are predefined for the multimedia function used).

6.6.3.1 Case 1: using the multimedia function f_1

The predefined multimedia objects represent a collection of a person's photos learned and assigned to the multimedia function f_1. We conducted a series of tests using, at each time, one of the three predefined multimedia objects (B_1, B_2 and B_3) under the 3 different profiles P_1, P_2 and P_3 respectively. The system captures 5 snapshots of the person and processes them using the multimedia function f_1 and an uncertainty threshold = 0.1 in P_1, P_2 and P_3. After, f_1 returns 5 scores representing the degree of similarity of the captured image with the predefined one. For each profile, we retrieve the highest result after applying the corresponding aggregation function to the 5 scores returned by the multimedia function f_1. The Table 6.4 shows the results in each profile with different predefined set of multimedia objects (B_1, B_2 and B_3). Using B_1 and P_1, the template T_4 based on the DS function returned the highest result in the 3 different profiles, whereas T_5 corresponding to the BN function returned almost the same values as the DS with a slight difference.

B_1	P_1	P_2	P_3
T_1	0.6	0.06	0.06
T_2	0.6	0.6	0.06
T_3	0.6	0.49	0.06
T_4	0.98	0.80	0
T_5	0.88	0.24	0
T_6	0.6	0.06	0.06

B_2	P_1	P_2	P_3
T_1	0.06	0.6	0.06
T_2	0.6	0.6	0.06
T_3	0.38	0.6	0.06
T_4	0.14	0.98	0
T_5	0.013	0.88	0
T_6	0.06	0.6	0.06

B_3	P_1	P_2	P_3
T_1	0.06	0.06	0.6
T_2	0.06	0.06	0.6
T_3	0.06	0.06	0.6
T_4	0	0	0.98
T_5	0	0	0.88
T_6	0.06	0.06	0.6

Table 6.4 Template tuning according to P_1, P_2 and P_3 using f_1

6.6.3.2 Case 2: using the multimedia function f_2

The predefined object here represents a location associated to the multimedia function f_2. As in case 1, we elaborate a series of tests using at each time one of the three predefined multimedia objects (B_1, B_2 and B_3) under the 3 different profiles P_1, P_2 and P_3 respectively. The system captures 5 snapshots of the location and processes them using the multimedia function f_2 and an uncertainty threshold = 0.1 in P_1, P_2 and P_3. The results are shown in Table 6.5. The template T_4 and T_5 returned the highest values in the 3 different profiles P_1, P_2 and P_3, whereas, T_1, T_2, T_3 and T_6 returned the same values approximately.

B_1	P_1	P_2	P_3
T_1	0.82	0.66	0.04
T_2	0.84	0.71	0.72
T_3	0.83	0.69	0.18
T_4	0.99	0.99	0
T_5	0.99	0.98	0
T_6	0.84	0.71	0.04

B_2	P_1	P_2	P_3
T_1	0.60	0.67	0.8
T_2	0.63	0.8	0.59
T_3	0.62	0.75	0.58
T_4	0.95	0.99	0.91
T_5	0.92	0.99	0.85
T_6	0.62	0.8	0.59

B_3	P_1	P_2	P_3
T_1	0.07	0.05	0.68
T_2	0.12	0.05	0.69
T_3	0.08	0.05	0.68
T_4	0	0	0.98
T_5	0	0	0.98
T_6	0.07	0.05	0.69

Table 6.5 Template tuning according to P_1, P_2 and P_3 using f_2

6.6.3.3 Uncertainty threshold tuning

We will calculate in the following the uncertainty threshold representing the difference between the returned results in each profile P_1, P_2 and P_3. For instance, the $\varepsilon = P_1 - P_2$ describes the value that could affect the aggregated result if captured images are taken in 2 different profiles P_1 and P_2. The Table 6.6 shows the calculated uncertainty thresholds between each profile for the specified predefined objects (B_1, B_2 and B_3). Using this uncertainty threshold calculation, one can adapt template usage based on the environments in which multimedia objects processing is taking place.

B_1	$\varepsilon_1 = P_1 - P_2$	$\varepsilon_2 = P_1 - P_3$
T_1	0.16	0.78
T_2	0.13	0.12
T_3	0.14	0.65
T_4	0	0.99
T_5	0.01	0.99
T_6	0.13	0.8

B_2	$\varepsilon_1 = P_2 - P_1$	$\varepsilon_2 = P_2 - P_3$
T_1	0.07	0.09
T_2	0.17	0.21
T_3	0.13	0.17
T_4	0.04	0.08
T_5	0.07	0.14
T_6	0.18	0.21

B_3	$\varepsilon_1 = P_3 - P_1$	$\varepsilon_1 = P_3 - P_1$
T_1	0.63	0.61
T_2	0.64	0.57
T_3	0.63	0.6
T_4	0.98	0.98
T_5	0.98	0.98
T_6	0.64	0.62

Table 6.6 Uncertainty Threshold calculation

6.7 Conclusion

Through this chapter, we showed how it is possible to bridge sensing and decision making using predefined templates in ambient intelligence environments. Templates grouping multimedia processing tools facilitate the manipulation of complex multimedia techniques which makes the process of retrieving knowledge from contextual information an easy task. In addition, we presented our uncertainty resolver in which we defined a set of adapted aggregation functions based on several probabilistic theories used in the literature, in order to reduce the uncertainty raised due to multimedia objects processing. The aim was to aggregate the set of values returned from several multimedia functions or sources into one relevant score which reduces the probability of faulty decisions. We elaborated also a set of experiments to demonstrate the efficiency of our aggregation functions when integrated in real-time processing and image capturing scenario. Several observations have been discussed through the results obtained.
In the near future, we intent to extend our approach to integrate several new aggregation functions (e.g. Artificial Neural Networks) in order to provide more efficiency in bridging the gap between sensing and decision making. Furthermore, we are currently implementing a prototype involving all the concepts provided here and willing to use it in real distributed application scenario(s).

References

1. J. Covington, M., J. Moyer, M., Ahamad, M. : Generalized Role-Based Access Control for Securing Future Applications. 23rd National Information Systems Security Conference, (pp. 16-19). Baltimore, MD, USA (2000)
2. Jean, K., Yang, K., Galis, A. : A Policy Based Context-aware Service for Next Generation Networks. 8th London Communication Symposium. London (2003)
3. Kumar, A., M. Karnik, N., and Chafle, G. : Context Sensitivity in role based access control. Operating Systems Reviews, **36**(3), 53–66 (2002)
4. Toninelli, A., Montanari, R., Kagal, L., Lassila, O. : A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments. International Semantic Web Conference (pp. 473-486). Athens, GA, USA: Springer (2006)
5. Wolf, R., Schneider, M. : Context-Dependent Access Control for Web-Based Collaboration Environments with Role-based Approach. MMM-ACNS 2003, pp. 267-278. St. Petersburg, Russia: Springer (2003)
6. Wullems, C., Looi, M., Clark, A. : Towards Context-aware Security: An Authorization Architecture for Intranet Environments. PerCom Workshops, pp. 132-137. Orlando, Florida, USA: IEEE Computer Society (2004)
7. Davis, M., Smith, M., Stentiford, F., Bamidele, A., Canny, J., Good, N., King, A., Janakiraman, R. . Using Context and Similarity for Face and Location Identification. 18th Annual Symposium on Electronic Imaging Science and Technology Internet Imaging. San Jose, California: IS&TSPIE Press (2006)
8. Mitchell, S., Spiteri, M. D., Bates, J., and Coulouris, G.: Context-Aware Multimedia Computing in the Intelligent Hospital. SIGOPS EW2000, the Ninth ACM SIGOPS European Workshop (pp. 13-18). Kolding, Denmark: ACM (2000)

9. Quinlan, J. R.: Induction of Decision Trees - Machine Learning. 1 (1986)
10. Poole, D.: Logic, Knowledge Representation, and Bayesian Decision Theory. Computational Logic, pp. 70-86. London, UK: Springer (2000)
11. Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press (1976)
12. Dey, A. K., Abowd, G. D., Salber, D.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human-Computer Interaction **16**, 97–166 (2001)
13. Dejene, E., Scuturici, V.-M., Brunie, L.: Hybrid Approach to Collaborative Context-Aware Service Platform for Pervasive Computing. Journal of Computers (JCP) , 40–50 (2008)
14. Wang, X., Zhang, D., Gu, T., Keng Pung, H.: Ontology Based Context Modeling and Reasoning using OWL. PerCom Workshops . Orlando, Florida, USA: IEEE (2004)
15. Bhatti, R., Bertino, E., Ghafoor, A.: A Trust-Based Context-Aware Access Control Model for Web-Services. Distributed and Parallel Databases , 83–105 (2005)
16. Hu, J., and C. Weaver, A.: A Dynamic, Context-Aware Security Infrastructure for Distributed Healthcare Applications. 5th Workshop on Pervasive Security Privacy and Trust (PSPT). MA, Boston (2004)
17. J. Covington, M., Fogla, P., Zhan, Z., Ahamad, M.: A Context-Aware Security Architecture for Emerging Applications. ACSAC 2002 (pp. 249-260). Las Vegas, NV, USA: IEEE Computer Society (2002)
18. J. Covington, M., Long, W., Srinivasan, S., K. Dey, A., Ahamad, M., D. Abowd, G.: Securing context-aware applications using environment roles. SACMAT 2001, pp. 10-20. Chantilly, Virginia, USA: ACM (2001)
19. Ardagna, C. A., Cremonini, M., Damiani, E., De Capitani di Vimercati, S., Samarati, P.: Supporting location-based conditions in access control policies. ASIACCS 2006, pp. 212-222, Taipei, Taiwan: ACM (2006)
20. IBM. QBIC - DB2 Image Extenders. Retrieved 02 16, from *http://wwwqbic.almaden.ibm.com* (2008)
21. Nepal, S., V. Ramakrishna, M.: Query Processing Issues in Image (Multimedia) Databases. International Conference on Data Engineering (ICDE), pp. 22-29, Sydney, Australia (1999)
22. Network Oracle Technology. Oracle Multimedia. Retrieved 11 09, from *http://www.oracle.com/technology/products/intermedia/index.html* (2007)
23. Lab, efg's Computer. Image Processing. Retrieved 01 02, from *http://www.efg2.com/Lab/Library/ImageProcessing/SoftwarePackages.htm*, (2008)
24. Press Java Community. Community Development of Java Technology Specifications. Retrieved 01 05, from *http://jcp.org/en/jsr/detail?id=135*, (2008)
25. Quinlan, J. R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers (1993)
26. Smach, F., Lemaitre, C., Miteran, J., Gauthier, J. P., Abid, M.: Colour Object recognition combining Motion Descriptors, Zernike Moments and Support Vector Machine. IEEE Industrial Electronics, IECON, pp. 3238-3242, Paris - France (2006)
27. P. Dempster, A.: A Generalization of the Bayesian Inference. Journal of Royal Statistical , 205–447 (1968)
28. An, X., Jutla, D., Cercone, N.: Privacy intrusion detection using dynamic Bayesian networks. 8th international conference on Electronic commerce: The new e-commerce: innovations for conquering current barriers, obstacles and limitations to conducting successful business on the internet, pp. 208 - 215, Fredericton, New Brunswick, Canada (2006)
29. Jemili, F., Zaghdoud, M., Ben Ahmed, M.: A Framework for an Adaptive Intrusion Detection System using Bayesian Network. ISI, pp. 66-70, New Brunswick, New Jersey, USA Computer Science (2007)
30. Chen, Y., Liginlal, D.: Bayesian Networks for Knowledge-Based Authentication. IEEE Transactions on Knowledge and Data Engineering , 695–710 (2007)
31. Yuan, Y., Shaw, M. J.: Induction of fuzzy decision trees (1995)