Chapter 14

Multimodal workbench for automatic surveillance applications

Noticeable developments have lately been achieved on designing automated multimodal smart processes to increase security in every-day life of people. As these developments continue, proper infrastructures and methodologies for the aggregation of various demands that will inevitably arise, such as the huge amount of data and computation, become more important. In this paper we introduce a multimodal framework with support for an automatic surveillance application. The novelty of the attempt resides in the modalities to underpin data manipulation as a natural process but still keeping the overall performance at high levels. At the application level the typical complexity behind the emerging distributed multimodal systems is reduced in a transparent manner through multimodal frameworks that handle data on different abstraction levels and efficiently accommodate constituent technologies. The proposed specifications includes the use of shared memory spaces (XML data Spaces) and smart document-centered content-based data querying mechanisms (XQuery formal language). We will also report on the use of this framework in an application on aggression detection in train compartments.

14.1 INTRODUCTION

The challenge to build reliable, robust and scalable automated surveillance systems has interested security people ever since the first human operated surveillance facilities came into operation. Moreover, since the bombings in London and Madrid in 2005, research in methods to detect potentially unsafe situations in public places has taken flight.

Given the size and complexity of the sensing environment surveillance systems have to cope with, including the unpredictable behavior of people interacting in this environment, current automated surveillance systems typically employ diverse algorithms (each focusing a specific feature of the sensor data), many sensors (with overlapping sensing area and able to communicate with each other through a network), and different types of sensors (to take advantage of information only available in other modalities).

It is our belief that in modern surveillance applications, satisfactory performance will not be achieved by a single algorithm, but rather by a combination of interconnected algorithms. In this chapter, we present a framework for automated surveillance designed to facilitate the communication between different algorithms. This framework is centered on the shared memory paradigm, the use of which allows for loosely coupled asynchronous communication between multiple processing components. This decoupling is realized both in time and space. The shared memory in the current design of the framework takes the form of XML data spaces. This suggests a more human-modeled alternative to store, retrieve and process data. The framework enhances the data handling by using a document centered approach to tuple spaces. All the data is stored in XML documents and these are subsequently received by the data consumers following specific XML queries. In addition, the framework also consists of a set of software tools to monitor the state of registered processing components, to log different type of events and to debug the flow of data given any running application context.

The remainder of this chapter is structured as follows. Section 14.2 starts with an overview of the related work, examining the existing approaches and motivate the key ideas of each approach. Then we give an overview of our framework and discuss its main building blocks. Section 14.4 shows the framework in action with an example of a surveillance application built on top of the framework. The application uses several cameras and microphones to detect unusual behavior in train compartments. Finally section 14.5 summarizes our contribution.

14.2 RELATED WORK

14.2.1 Video based approaches in automated surveillance research

The bulk of existing work on automated surveillance, has largely concentrated on using only video. Video based surveillance related algorithms have been extensively investigated [6]. The underlying algorithms consist of methods ranging from simple background extraction algorithms to more complex methods such as dynamic layer representations [11] and optical flow methods [7].

Depending on the characteristics of the sensing environment more specific methods have been employed. The performance of face detection algorithms [3], for example, depends on the size of the surveillance area, whereas the different methods for people or (more generally) object detection and tracking depend on the amount of occlusion and the motion characteristics of the objects involved. Even in visual event detection, different methods, video event graphs [22] and HMMs [26], have been developed.

Some researchers approach the surveillance problem with special kinds of visual sensors and their specific algorithms. For example in [8] the authors integrate normal camera, infrared (IR) and Doppler vibrometers (LDVs) in their multimodal surveillance system for human signature detection.

14.2.2 Audio based approaches in automated surveillance research

As sound/speech is not always present, it's impractical to do continuous sound based tracking. Therefore, sound based surveillance algorithms are usually sound event detectors or recognizers. These algorithms usually consist of a training phase in which various features are extracted from training sound signals to obtain characteristic acoustic signatures of the different types of events. During operation, a classification system matches sound signals against the trained acoustic signatures to detect events [19][17][18]. [12] adapt a multi level approach in which audio frame are first classified into vocal and nonvocal events. Then a further classification into normal and excited events is performed. The events are modeled using a Gaussian Mixture Model with optimized parameters for four different audio features.

14.2.3 Multimodal Audio-video based approaches

Sound based algorithms are most effective when used in combination with sensors from different (in particular the visual) modalities. Audio can be used complimentary to help resolve situations where video based trackers loose track of people due to occlusion by other objects or other people. The combined approach yields better and more robust results as demonstrated by different researchers using decentralized Kalman filters [13], particle filters [14] and Importance Particle Filters [16]. In general, multimodal approaches of surveillance applications consists of unimodal, modality specific, low level feature extraction algorithms and higher level multi-modal fusion algorithms. For example in [9] the authors use Kalman filtering to combine standalone audio and video trackers. Other techniques used in higher level fusion are the Bayesian networks [10] [2] [4], rule based systems [1] and agent based systems. A different approach is adopted in [15] where a system for tracking of moving object was developed, using a probabilistic model describing the joint statistical characteristics of the audio-video data. Typical of this approach is that fusion of the data is done at a low level, exploiting the correlations among the two modalities. The model uses unobserved audio and video variables to describe the observed data in terms of the process that generates them.

14.2.4 High level interpretation

At a higher level, automated surveillance research is focused on semantic interpretation of the events in their spatial and temporal relationships. In [20] an automatic surveillance system is discussed, that performs labeling of events and interactions in an outdoor environment. It consists of three components - an adaptive tracker, an event generator, which maps object tracks onto a set of pre-determined discrete events, and a stochastic parser. The system performs segmentation and labeling of surveillance video of a parking lot and identifies person-vehicle interactions. In [5] human behavior models are used to obtain interpretation.

Generally, high level approaches are heavily dependent on the context in which the system is applied. There are at least two projects where extensive research has been conducted in applying video and audio

processing algorithms for improving passenger safety and security in public transport systems. In the PRISMATICA project, [2] developed a surveillance system for railway stations, designed to integrate different intelligent detection devices. In the PRISMATICA system, detection of events is done in geographically distributed visual, audio and other types of devices and using different algorithms. Bayesian networks are used for presenting and fusing the data by these devices. In the ADVISOR project [1], a surveillance system was developed for detecting specific behavior (such as fighting or vandalism) in metro stations. Different methods have been defined to compute specific types of behaviors under different configurations. All these methods have been integrated in a coherent framework.

14.2.5 Frameworks

In most related research, the focus has been on the algorithms to achieve the automated surveillance. In most cases, the overall framework to make the algorithms work with each other is added as an afterthought. From a system engineering perspective, this approach is far from optimal, as very important functions such as communication, resource management (e.g. handling distributed data sources, data sharing) and exception handling (if a sensor might break down) rely on a good overall framework.

In recent years application specific frameworks based on emerging technologies such as peer-to-peer environments, message based middle ware and service oriented approaches have been developed. Most use XML or XML based technologies to support data manipulation and to facilitate component interaction. XMIDDLE, for example, is a mobile computing middleware using XML and Document Type Definition (DTD) or Schema to enhance data with typed structure and Document Object Model (DOM) [24] to support data manipulation. It further uses XPath [25] syntax to address the data stored in a hierarchical tree structured representation. Other examples are Jini and Java Media Framework (by Sun Microsystems), Vinci [21], and HP Web Services Platform and e-Speak (by Hewlett Packard). Finally, there have been frameworks developed for multimodal fusion applications e.g. [47] [23]. Table 1 gives a comparison of existing frameworks. We have tried to make an overview of the frameworks used by the

different researches, and briefly indicate their characteristics. We omit detailed descriptions due to lack of space.

Table 14.2-1

Comparison of existing frameworks. The comparison is based on: modality dependency (1), applied for surveillance applications (2), scalability (3), performance (4), transparency (5), modularity (6), repository services (7), coordination services (8) and security services (9). The qualifications +, 0, -, and ? indicate good, neutral, bad and unknown respectively.

Framework	Ref.	1	2	3	4	5	6	7	8	9
XMIDDLE		0	-	+	0	+	+	+	+	+
JMF/Jini	[40]	0	-	+	0	+	+	+	+	+
Vinci	[21]	0	-	+	+	+	+	+	+	-
E-Speak		0	-	+	0	+	+	+	+	+
iROS	[23]	+	?	?	-	+	+	+	+	-
ADVISOR	[1][10]	+	+	+	+	+	0	-	-	-
PRISMATICA	[2]	+	+	+	+	+	+	-	-	-
KNIGHT	[4]	-	+	?	+	?	0	-	-	-

14.3 GENERAL MODEL DESCRIPTION FOR THE MULTIMODAL FRAMEWORK

The multimodal framework being described in this paper is centered on the shared memory paradigm. It introduces a novel technique in the way data is handled by different purpose data consumers. Comparing with the traditional way implying direct connections between the system components each connection having its own data format, the new approach suggests a more human-modeled alternative to store, retrieve and process the data. The data is conferred an underlying structure that complies with eXtended Markup Language (XML) [34]. The shared memory in the current design of the multimodal framework takes the form of XML data spaces.

The use of shared memory allows for loosely coupled asynchronous communication between multiple senders and receivers. The communication decoupling is realized both in time and location. The specification fully complies with the requirements of data manipulation in a multi data producer/consumer context where the availability of data is time-dependent and some connections might be temporarily interrupted.

The information is structured in documents using XML standard. XML Schema [35] is used to validate each existing XML document prior to extracting the meaningful information. Furthermore, the binary data can be easily interchanged via XML documents after converting it using XML MIME protocol. The multimodal framework also consists of a set of software tools to monitor the state of registered processing components, to log different type of events and to debug the flow of data given any running application context.

Depending on the type of application to be built on top of the multimodal framework, a routing algorithm has been design to manage the data transfer among existing shared memories on different physical networks. This capability is highly required commonly for multimodal applications that involve distinct wireless devices. Considering the study case of an automatic surveillance application, wireless devices such as PDAs or mobile phones equipped with video camera can communicate with the system core to send useful video data.

The framework specifications solely emphasize the presence and role of all its components through existing technologies and standards and not on the implementation details. Yet several proposed technologies present a certain degree of freedom in some functional aspects for the implementation phase. Although the multimodal framework has been designed by taking into consideration the further development of an automatic surveillance oriented application, it can be successfully adopted as basis for any kind of complex multimodal system involving many components and heavy data exchange. By making use of the framework specifications a common description of possible processing components along with their interconnections for a surveillance application is provided as ground for eventual specific examples that may be given throughout the paper. Because the workbench implementation itself relies on the philosophy of shared XML data spaces, a special attention is given to examples on how to integrate the two underlying technologies for modules and XML data management.

The examples aim at studying the usability and extensibility of concepts in formulating proper data and command maneuvers to ensure a logic and natural data flow through the network of data processing nodes.

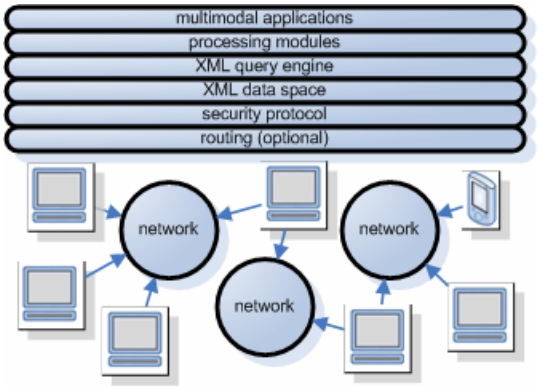

Figure 14.3.1 *The multimodal framework diagram*

The algorithms in the illustrated processing components related to the example of automatic surveillance application are not described in full detail due to the space and topic considerations. Indeed they range to broad research study areas from different computing fields and barely any standard exists to favor one method to the other. In restricted cases some algorithms are employed for exemplification purposes though.

One distinct remark concerning the multimodal framework is that it should support platform independent interconnection of the processing modules. This requirement is essential for heterogeneous working environments while it also leaves open space for what programming languages, tools and technologies to be used at the implementation phase. An overview of the current multimodal framework is shown in the diagram in Figure 14.3.1.

If the processing modules are envisaged as services in the multimodal workbench, then a multimodal application specifies, along with the service set, a working plan on how the services are used so as to get the data through different abstraction level to the desired semantic information. The multimodal framework defines a way to represent the work plan through a Monitor application and uses that to get a detailed state analysis over its fulfillment. Each service registered in the framework publishes its input and output XML formatted specifications using Web Services Description Language (WSDL) [33].

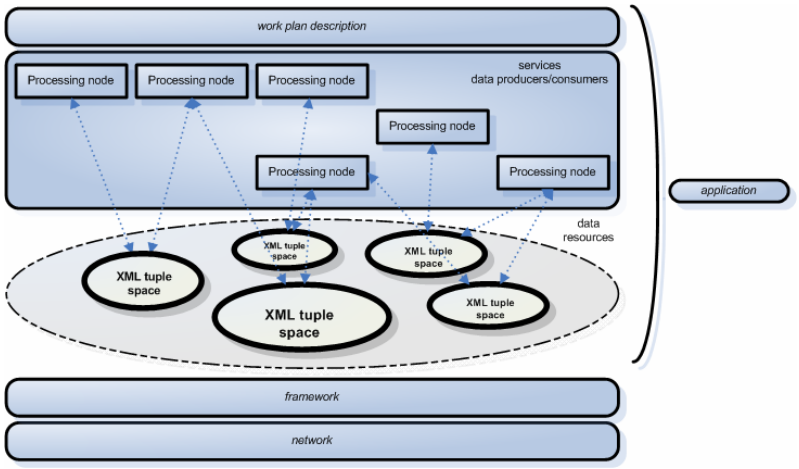

Figure 14.3.2 *The service oriented overview on the multimodal framework.*

The service oriented overview is given in Figure 14.3.2. Related technologies to define distributed services in Distributed Object Environments (DOE) include Jini [40] defined by Sun Microsystems in the late 1990's, Service-Oriented Architecture (SOA), Common Object Request Broker Architecture (CORBA) from OMG and Distributed Component Object Model (DCOM) from Microsoft.

14.3.1 XML data spaces

In the proposed multimodal workbench, the processing of data takes place in a distributed manner. Several technologies like RPCs, Remote Method Invocation (RMI) or Common Object Request Broker Architecture (CORBA) that have been researched in the past to overcome the problems imposed by the distributed computing environment still present some inaccuracies. Due to its capabilities to solve the typical problems regarding synchronization, persistence and data communications the tuple spaces algorithm has been extensively researched and used ever since its introduction by Yale University in the mid 1980's. The goal of the original Linda system was to achieve coordination within various parallel applications. The greatest advantage of the tuple spaces is the decoupling of sender and receiver components in both time – by removing the existence simultaneity restriction, and space – by removing the address information requirement. One component being part of the shared memory space needs to know neither the location of the component it plans to send data to nor anything about its availability.

Later Linda versions include Melinda, a multiple data space Linda based algorithm, Limbo for adaptive mobile applications, Lime and eLinda that specifies fully distributed tuple spaces and an additional form of output operation. The other eLinda extensions regard the Programmable Matching Engine (PME) as a more flexible criterion for tuple addressing and support for multimedia through Java Media Framework (JMF). JavaSpaces from Sun and TSpaces [36] from IBM are recent Linda variants that store and exchange data in form of objects.

JavaSpaces uses transaction term to denote a set of tuple space operations that can be rolled back in some faulty cases, and lease term to mention the automatic removal of tuples after the time expiration.

TSpaces allows for programming language independent services described either using WSDL or proprietary description languages (IDLs) and for a general purpose mechanism to associate tuplespace events with actions via specific rules. sTuples [42] proposes an architecture for Semantic Tuple Spaces that extends a secure communication framework to support a description-logic reasoning engine and a web ontology language.

A recent improvement for the tuple-spaces algorithms allows for XML-documents handling. A few implementations already exist as xSpace [31], Rogue - Ruple [37] or XMLSpaces.NET [38]. xSpace advances xSpaces Query Language, a XPATH syntax like querying language using both content and structure information with support for both XML Schema and DTD. XMLSpaces.NET implements the tuple spaces algorithms on the .NET platform.

Our implementation of the multimodal workbench is an adapted version of Ruple XML spaces. Ruple Alpha V2 release includes XML Spaces, HTTP and Simple Object Access Protocol (SOAP) for data accessibility, MIME attachments and a subset of XML Query Language (XQL) [31] for document retrieval.

The design of our XML space follows the simplicity criterion so as to provide the multimodal workbench with fast data manipulation and in the same time to present high enough complexity for data

storage and retrieval. All the documents written in spaces supported by the multimodal framework are in XML format. The four basic operations Figure 14.3.3 on XML data spaces supported by the multimodal middleware are: *write* – for sending an XML document that have an expiration time to a XML space, *read* – for retrieving an XML document from an XML space based on an XML query, *readMultiple* – for retrieving a set of XML documents based on a XQuery XML query and *take* – for reading and deleting an XML document that matches an XML query.

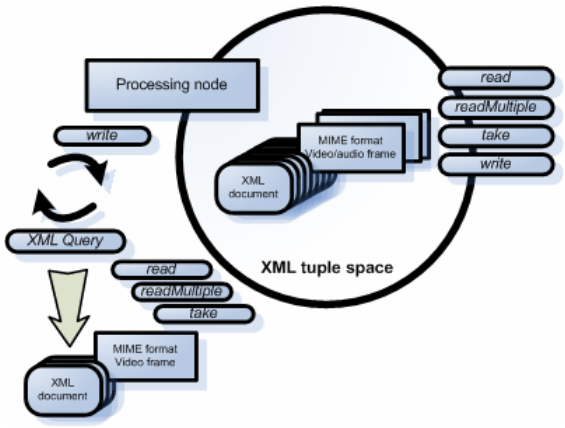

Figure 14.3.3 *The basic operations used to handle the XML documents in XML tuple spaces.*

The case of querying XML documents from several XML tuple spaces is illustrated in Figure 14.3.4. Such an operation demands for distinct connections to each of the XML tuple spaces involved in the query. After the retrieval of the documents from the data space using the basic tuple operations, the proper document is selected by using a query expression. This query specifies how the XML document to be taken should look like in terms of specific attribute values. The last step is the extraction of audio/video data from the queried XML document and the framework base64 decoding used for generating the original audio/video data.

For an application context where several processing services are working simultaneously as data consumers and produces, there can be assumed the existence of a set of XML spaces assuming the integration of the different nodes in the multimodal framework. Our framework does not have any restrictions for the number of XML data spaces. Moreover, given a distributed running context, the

current XML data flows are optimized through different data spaces enabling for high-speed data propagation in the services network. A general integration of XML spaces is presented in Figure 14.3.5. Every two data spaces are asynchronously connected through at least one common node in the network.

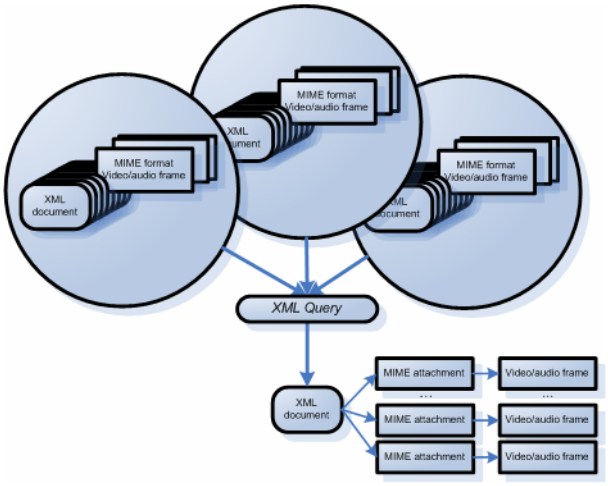

Figure 14.3.4 *Audio/video data retrieval based on XML query from multiple XML tuple spaces*

The connecting node can be part of different physical networks and so acting as a bridge between XML tuple spaces lying in distinct networks. This case is the first case of Figure 14.3.5 (Physical Network Layer I) where the three XML tuple spaces reside in three different physical networks (*Net1*, *Net2* and *Net3*). As can be seen there are processing nodes (*PN1- PN2*, *PN8 - PN9* and *PN11 - PN12*) that co-exist on the same hardware device and are connected to the same XML tuple space. The nodes with no direct visibility can still exchange data through a sequence of interconnected XML spaces.

The second case (Physical Network Layer II) illustrates the example of a unique physical network (*Net*) that supports all the XML spaces in the multimodal framework. Our implementation of XML tuple spaces respects the general specifications followed also by other papers. Along with the implementation of the four basic XML space operations, our framework exports certain functions to the processing modules and the Monitor application in the multimodal workbench.

More exactly, the *creation of a new space* operation allows for initiating a new empty XML tuple space by both the services and the Monitor application. In the same way it is possible to remove a XML

space from the workbench with the *deletion of a space* operation. In this case all the XML documents that existed up to the time of deletion are also deleted. Any processing module can subscribe to an XML data space by calling the *registration to a space* operation.

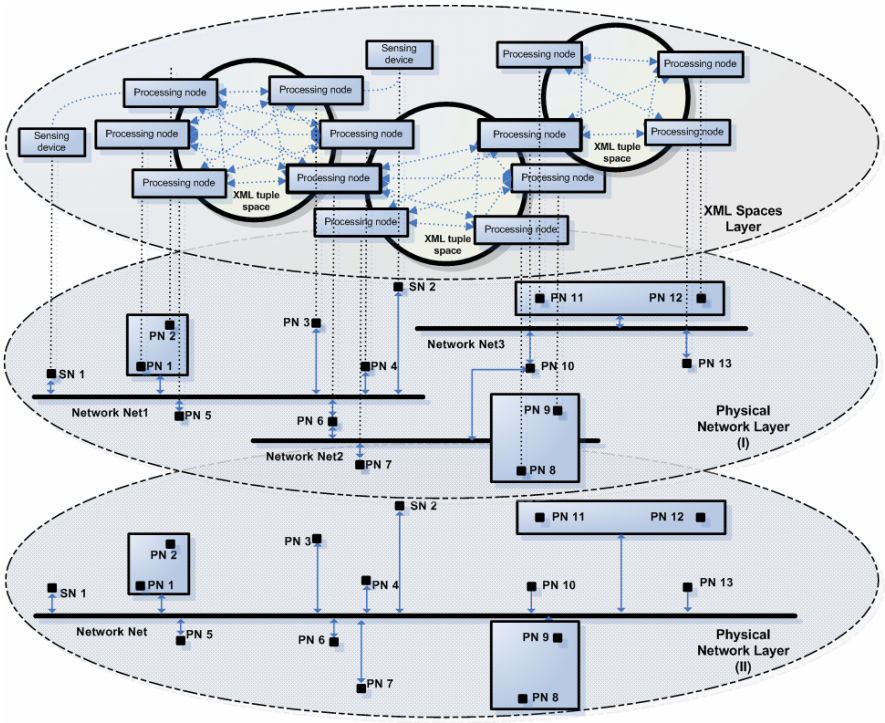

Figure 14.3.5 *Connections between multiple data spaces and the distribution of nodes on the physical network.*

The reverse operation is also supported for un-registering from a XML space through the *un-registering from a XML space* operation. The set of XML tuple spaces can be searched for the existence of a certain XML space by using the *searching for a space* operation. The function provides several characteristics regarding the XML space as the time of XML space creation, the entity that created it, the maximum capacity and the number of XML documents in the space and performance related measures as the frequency and the workload of operations applied on the space. The list of all the existent spaces can be retrieved using the *retrieve the XML space list* operation.

14.3.1.1 XML document lease

The framework automatically checks for the availability of the documents as it is specified by the life-span parameter in the XML document lease. Before writing to a XML data space, each XML document is assigned a lease specifying an expiration time. The parameter determines for how long the

document is available in the XML data space. Every time the XML document list is checked and a XML document is found as having the expiration time up, it is removed from the XML document set of the XML data space. The XML space assures for an efficient flexible document exchange between the sender node and the destination node. Occasionally connected devices can provide data to the multimodal framework and also to receive processing results.

14.3.1.2 Multimedia data support

An XML document may include audio/video data as attachment. The framework includes functions Figure 14.3.6. The base64 [41] binary to text encoding scheme is used to convert an audio or video frame byte sequence to a sequence of printable ASCII characters. This operation allows for the integration of binary data into text-based XML files. The reverse of the encoding is requested at the retrieval of a XML document from a XML tuple space, prior to the actual audio/video data processing.

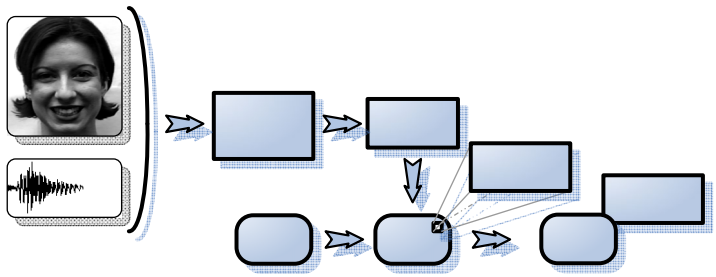

Figure 14.3.6 *The transformation of binary data to base64 formatted attachments.*

14.3.1.3 The model of XML spaces at different abstraction levels

Due to the multimodal nature of the target application the current framework involves operations that fuse data of different types. In order to make room for a proper classification of data fusion operations accommodated by the multimodal framework, the types of data on different abstraction scales have to be introduced. On the first level of abstraction is the raw data (feature level). This represents unchanged data as it is received from the input-sensing devices such as video frames from video cameras or audio signals

from the microphones. In our implementation of the workbench, the low level data are stored in *Raw data XML spaces* (Figure 14.3.7).

The second type assumes an intermediate level which includes all the data derived from input, raw data. It represents all kind of variables that are to be used as inputs to other processing modules yet do not have a self contained meaning to the application-context from the user point of view. As for examples, the facial features as the location of facial characteristic points or other computed parameters as distances or angles between them or the phoneme set in the case of speech analysis are considered to belong to the class of intermediate data. The intermediate abstraction level data are stored in *Intermediate data XML spaces* (Figure 14.3.7).

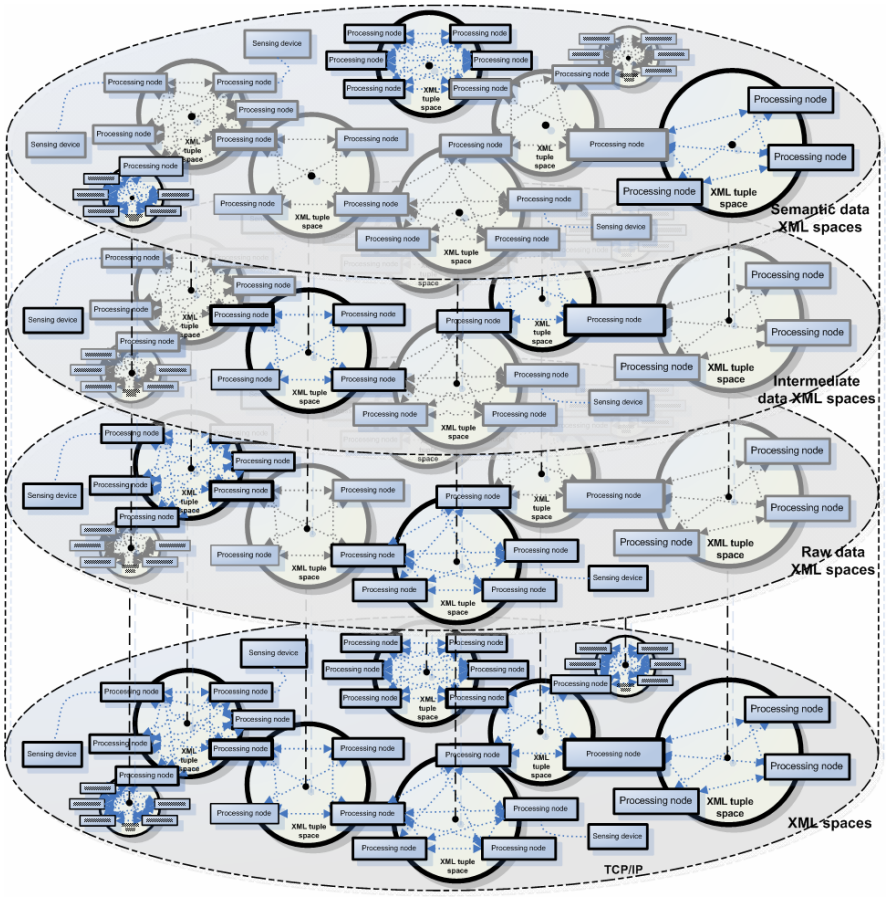

Figure 14.3.7 *The transformation of data to higher levels of abstraction through different XML Spaces.*

The third type is the high-level, semantic data. This class contains highly processed data extracted by following close analysis on raw, intermediate or other high-level data. These data have a high level of abstraction and are meant to explain the state and behavior of various entities in the application context. The high level data are stored in *Semantic data XML spaces* (Figure 14.3.7). To enumerate some examples these include the number of faces or persons from the view along with meaningful information as the emotions shown by everyone, the talk, the gaze for each person as well as information related to other existing objects or external events as background voices.

According to the levels of abstraction of data on which the fusion is done, there are three types of data fusion transformations through which the processing modules in the multimodal framework are structured: low-level, intermediate and high level data fusion. The low-level data fusion implies the operations on raw data directly received from the sensing devices as video cameras or microphones and the output represents intermediate or high-level data on the abstraction scale. The intermediate data fusion operates on either intermediate-level or both low-level and intermediate-level data. The high-level data fusion takes into account all the operations considering high level data or combinations of high-level with intermediate and low level data. The processing modules implying fusion of data at a certain abstraction level are depicted in Figure 14.3.7 as *Processing nodes* that are active at a certain XML space layer denoting an data abstraction layer and are connected to other XML spaces from other layers.

14.3.2 Querying data from XML data spaces

The video/audio data as well as other derived information within the multimodal framework is XML document-centered and the retrieval is based on XML queries. Query languages present an efficient way to support content based information extraction and updates. Prior to running the processing on data the registered modules perform such a query on a specific XML shared data. The query is generated according to the specifications of XQuery format. XQuery is a typed, functional language for querying XML documents. It is currently developed by the XML Query Working Group of the World-Wide Web Consortium (W3C). The XQuery specification set provides a high-level description of the XML syntax

for a query as well as the underlying semantics. XQuery expressions consist in a set basic building blocks defined as Unicode strings containing keywords, symbols and operands.

There are several XML-related standards that XQuery shares common interoperability with. To mention a few, XML standard itself, XML Schema and XPath are highly dependable. Along with other requirements, the XPath describes how to manipulate the XML document content generated in correspondence with the associated XML Schema. The XML Schema contains all the necessary information on how the XML document is formatted. Similar ways of specifying the structure of an XML document are Document Type Definition (DTD) included in the original XML specification or Document Structure Description (DSD).

The requirements of data manipulation within our multimodal framework specify the presence of XML documents along with the XML Schema descriptions for each shared data space. This is a requirement for the multimodal framework and not necessarily for XQuery as the W3C specifications state that the existence of XML schema is not mandatory.

A major advantage of using XQuery is the FLWOR expression providing iteration and binding of variables to intermediate query representations. The XML document is parsed by an XML parser that generates an XML Information Set. In our implementation of the multimodal workbench we use Xerces [49] for parsing XML files using DOM/SAX parser. After parsing, the document is further validated against an XML Schema and the result is an abstract structure called Post-Schema Validation Infoset (PSVI). In our workbench implementation, an interface makes the connection between variables and functions in the framework and in XQuery expressions through the XQuery Expression Context (Figure 14.3.8).

The XML Schema is retrieved from the same tuple space as the one containing the XML documents. If a XML document does not comply with its XML schema an error is raised as it may contain flawed formatted data and further processing is no longer performed.

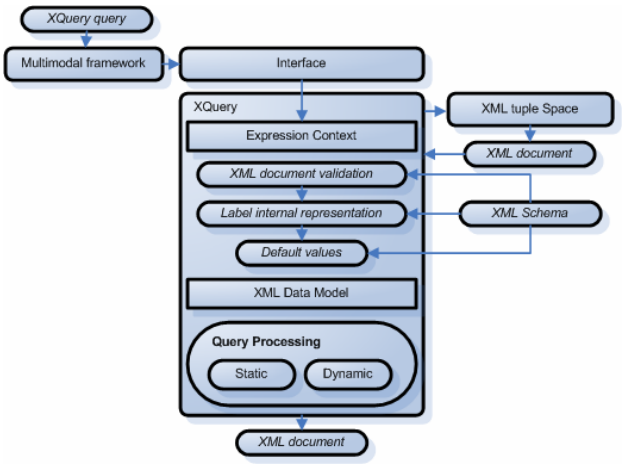

Figure 14.3.8 *interface between the multimodal framework and the XQuery.*

Once the document is checked to determine whether it has the correct format an internal representation is generated and labeled as indicated by the same schema. The third step that makes use of document schema focuses on possible missing attributes to assign default values.

The XML Data Model (XDM) resulting from the Information Set or PSVI undergoes a sequence of XQuery query processing including static and dynamic evaluation phases to determine the values of the inner expressions.

14.3.3 General description model of the multimodal workbench for the automatic surveillance application

The context awareness of our automatic surveillance application is generated by applying a set of data processing on input raw data received from the sensing devices as cameras and microphones. Figure 14.3.9 depicts a typical diagram of the necessary processing modules for supporting the automatic surveillance application. The processing modules range from face, gaze, gesture to sound, sound location, speech recognition and emotion analysis and aim at extracting useful information from audio and video data at different levels of abstraction.

The *Behavior recognition* module has the role of detecting any type of aggression that takes place in the scene being analyzed. The analysis focuses on each person and also on the groups by considering information semantic information related to the gestures, text from the conversations, emotions shown by each person and sound patterns as breaking glass or shouting. The aggression related high level data are

stored in XML data space '*scene.person.semantic.aggression*'. The information stored in this XML space is of a great importance for the feedback of the entire surveillance session. The actual feedback is supported by two types of actions. The first type represents the set of actions to be followed when a certain kind of aggression is detected by the system. It can take the form of video recording the whole session or generating SMS, email or other notifications. The relevant information is stored in XML space '*scene.systemFeedback*'.

The second type represents the actions concerning the commands to the Pan Tilt Zoom PTZ cameras. Along with sound, sound location, gaze and face, the aggression information is further on taken into account for determining the region of interest for surveillance. The processing takes place in *Focus* module and consists in a prediction of the system over the next possible places in the scene where aggression might take place.

A typical example of aggression scene assumes the detection of a sound event that is captured and with the relevant information stored in '*audio.semantic.sound*' space. Based on relevant information from the XML space '*scene.action.*focus', the system feedback is provided as motor commands to the video cameras to focus on that part of the scene. At the next processing moment the aggression actions are captured and interpreted by the system and eventually the specific feedback is generated according to the steps to be followed in such contexts. Listing 14.3-3 illustrates typical examples of querying data from tuple space '*video.frames'* that contains an XML document storing video frames received from multiple video cameras. The XML document has the XML Schema as indicated in Listing 14.3-1.

The query Q1 retrieves an XML formatted result comprising all the video frames as they are stored in the original document. The second query Q2 introduces a video frame selection based on two constrains pointing to data frames of type "*jpeg*" received from camera "*camera1*". The resulting partial XML formatted document contains only the XML element '*data*' out of each compound element '*frame*'. The query Q3 represents an alternative way of writing the query Q2, their results being the same. The last example Q4 iterates through all the video frames of type "*jpeg*" that are received in the last 60 seconds

from video camera "*camera1*". For each such video frame it calls a face detection function declared locally.

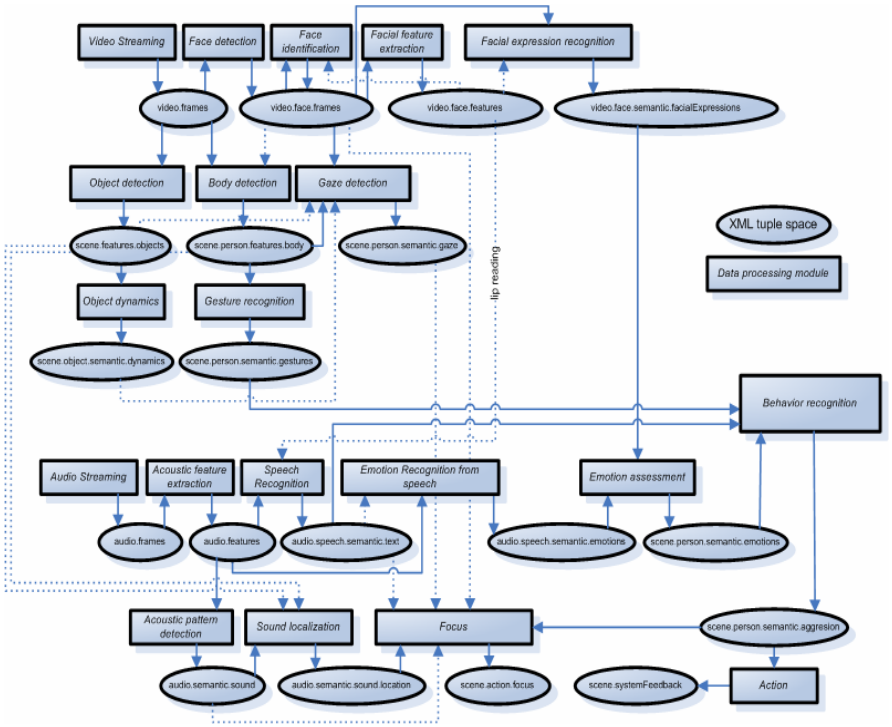

Figure 14.3.9 *The typical diagram of processing modules and shared data spaces for an automatic surveillance application.*

The face detection routine is managed by our multimodal workbench as a service that has the WSDL specification as exemplified in Listing 14.3-2. The connection between the published processing routine that is represented using *Multimodal Framework* component in the diagram (Figure 14.3.8), and the *XQuery* XML query processor – is realized through the component *Interface*. The interface assumes an optimal matching and translation of the input and output parameters of the routines existing in the workbench and those in the XQuery expressions.

Depending on the specific requirements for the routines of each data processing module, the type system of XQuery/XPath Data Model (XDM) may be augmented in an actual implementation to enhance the use of data variables of distinct types (i.e. face variables). Our implementation of the face detection

routine published by the *Face detection* module is based on Adaboost algorithm with Relevance Vector Machine (RVM) [45] as a weak classifier using Viola&Jones features.

The output of the query for retrieving the data for face detection may be an XML document satisfying XML Schema in Listing 14.3-4 that contain data on all the detected faces in the ascending order of the recording time.

```
<xs:schema xmlns="http://www.w3.org/2001/XMLSchema">
<xs:element name="videodata" type="videodata-type">
<xs:complexType name="videodata-type">
 <xs:sequence>
    <xs:element name="frame" type="frame-type" minOccurs="1" maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="frame-type">
 <xs:sequence>
    <xs:element name="cameraID" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="time" type="xs:time" minOccurs="1" maxOccurs="1"/>
    <xs:element name="width" type="xs:integer" minOccurs="1" maxOccurs="1"/>
    <xs:element name="height" type="xs: integer" minOccurs="1" maxOccurs="1"/>
    <xs:element name="dataType" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="data" type="xs:string" minOccurs="1" maxOccurs="1"/>
 </xs:sequence>
 <xs:attribute name="frameID" type="xs:string" use="required"/>
</xs:complexType>
</xs:schema>
```

Listing 14.3-1 *XML Schema for tuple <video.frames>.*

```
< wsdl:definitions>
                <xsd:element name="faceDetection">
                  <xsd:complexType>
                    <xsd:sequence>
                                <xsd:element name="width" type="string"/>
                                <xsd:element name="height" type="string"/>
                                <xsd:element name="dataType" type="string"/>
                                <xsd:element name="data" type="string"/>
                    </xsd:sequence>
                  </xsd:complexType>
                </xsd:element>

                < wsdl:message name="getWidthRequest">
                                <wsdl:part name="width " type="xs:unsignedShort"/>
                </wsdl:message>

                < wsdl:message name="getHeightRequest">
                                <wsdl:part name="height" type="xs:unsignedShort"/>
                </wsdl:message>

                < wsdl:message name="getDataTypeRequest">
                                <wsdl:part name="dataType" type="xs:string"/>
                </wsdl:message>

                < wsdl:message name="getDataRequest">
                                <wsdl:part name="data" type="xs:string"/>
                </wsdl:message>

                <wsdl:message name="getFaceResponse">
                                <wsdl:part name="face" type="faceDetection"/>
                </wsdl:message>

                <wsdl:portType name="FaceDetectionLib">
                                <wsdl:operation name="detectFaces">
                                                <wsdl:input message="getWidthRequest"/>
                                                <wsdl:input message="getHeightRequest"/>
                                                <wsdl:input message="getDataTypeRequest"/>
                                                <wsdl:input message="getDataRequest"/>
                                                <wsdl:output message="getFaceResponse"/>
                                </wsdl:operation>
                </wsdl:portType>
                :
</ wsdl:definitions>
```

Listing 14.3-2. *Example of WSDL statement for Face Detection processing module*

```
Q1: doc("video.frames.xml")/videodata/frame

Q2: doc("video.frames.xml")/videodata/frame[cameraID="camera1" and dataType="jpg"]/data

Q3:
for $video_frame in doc("video.frames.xml")/videodata/frame
where $video_frame/cameraID="camera1" and dataType="jpg"
return $video_frame/data

Q4:
<faces>
{
        ...
        let $current_time:= current-time()
        for $vf in doc("video.frames.xml")/videodata/frame
        where xs:time($vf/time)>$current_time-60
        order by xs:time($vf/@time)
        return
                <face faceID="{local:getFaceID($vf)}">
                    <personID>unidentified</personID>
                    $vf/cameraID
                    $vf/time
                    local:detectFaces($vf/width, $vf/height, $vf/dataType, $vf/data)
                </face>
}
</faces>
```

Listing 14.3-3 *Examples of using XQuery language to retrieve data from tuple <video.frames>.*

```
<xs:schema xmlns="http://www.w3.org/2001/XMLSchema">
<xs:element name="faces" type="faces-type">
<xs:complexType name="faces-type">
 <xs:sequence>
    <xs:element name="face" type="face-type" minOccurs="1" maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="face-type">
 <xs:sequence>
    <xs:element name="personID" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="cameraID" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="time" type="xs:time" minOccurs="1" maxOccurs="1"/>
    <xs:element name="width" type="xs:integer" minOccurs="1" maxOccurs="1"/>
    <xs:element name="height" type="xs:integer" minOccurs="1" maxOccurs="1"/>
    <xs:element name="dataType" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="data" type="xs:string" minOccurs="1" maxOccurs="1"/>
 </xs:sequence>
 <xs:attribute name="faceID" type="xs:string" use="required"/>
</xs:complexType>
</xs:schema>
```

Listing 14.3-4 *XML Schema for tuple <video.face.frames>.*

The example in Listing 14.3-7 shows the emotional information selection from both video and audio data for refining the assessment of emotions for the persons in the scene. Each audio frame is synchronized with the video frames (Figure 14.3.10) with respect to the individual recording timestamps. The result of the script contains the time ordered emotion information from audio and video structured using a tag to identify the human subject of the emotional analysis. In terms of XQuery specifications, the operation is a join on partial XML data from the two XML documents. The XML document storing emotional information about faces is stored in XML space '*video.face.semantic.facialExpressions'* and has the XML Schema as illustrated in Listing 14.3-5.

Our implementation for the facial expression recognition routines published by the *Facial expression recognition* module includes RVM and Support Vector Machines (SVM) [44], and Bayesian Belief

Networks (BBNs) [43] using a Face Model consisting of distance and angle features between typical Facial Characteristic Points (FCPs) on the human face. By using the Monitor application at runtime, the multimodal workbench exposes all the available services and it offers the possibility to choose which face detection module/service will be used for the current analysis session.

The XML document with emotional information on speech is stored in XML space '*audio.speech.semantic.emotion'* and has the XML Schema as in Listing 14.3-6. In our multimodal workbench, the implementation of the emotion recognition from speech routine published by the *Emotion recognition from speech* module is based on GentleBoost classifier [46] using an optimal utterance segmentation.

A distinct type of emotion recognition is the assessment of the stress level. The implementation of the stress recognition system [48] is realized as an embedded component of the *Emotion recognition from speech* module.

The generated XML document represents an intermediate result and constitutes the input of the *Emotion Assessment* processing module that stores the final emotion related information for each person in the scene into the XML data space '*scene.person.semantic.emotions*'.

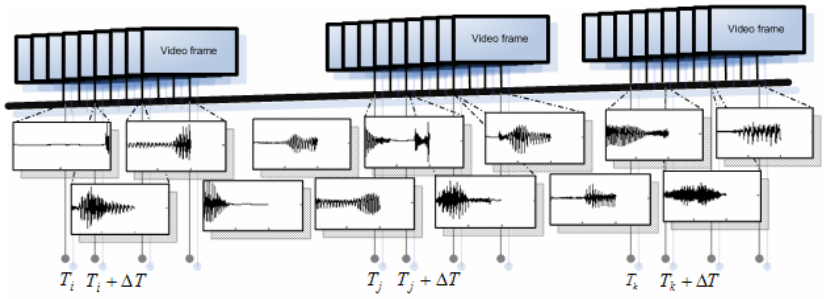

Figure 14.3.10 *The time span correspondence between video and audio frames.*

The actual routine published by the *Emotion Assessment* processing module would either remove the ambiguity in the case of contradictory data or would increase the result confidence in the case of semantic agreement. Though the semantic fusion case can be naturally sketched as a distinct module located in

*Semantic emotion recognitio*n module, various designs present it as part of either *Facial expression recognition* module or *Emotion recognition from speech* module, depending on the algorithms used and the aim of the analysis.

Our implementation of the routine for fusing audio and video semantic emotion data makes use of dynamic Bayesian Belief Networks (BBNs) to model the human emotions and to catch their inner temporal dependencies in a probabilistic framework.

```
<xs:schema xmlns="http://www.w3.org/2001/XMLSchema">
<xs:element name="video" type="video-type">

<xs:complexType name="video-type">
 <xs:sequence>
    <xs:element name="frame" type="frame-type" minOccurs="1" maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>

<xs:complexType name="frame-type">
 <xs:sequence>
    <xs:element name="personID" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="time" type="xs:time" minOccurs="1" maxOccurs="1"/>
    <xs:element name="emotion" type="emotion-type" minOccurs="1" maxOccurs="1"/>
 </xs:sequence>
 <xs:attribute name=" frameID" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="emotion-type">
 <xs:sequence>
    <xs:element name="happiness" type="xs:integer" minOccurs="1" maxOccurs="1"/>
    <xs:element name="sadness" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="fear" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="disgust" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="surprise" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="anger" type="xs:string" minOccurs="1" maxOccurs="1"/>
 </xs:sequence>
</xs:complexType>

</xs:schema>
```

Listing 14.3-5 *XML Schema for video.face.semantic.facialExpressions.xml.*

```
<xs:schema xmlns="http://www.w3.org/2001/XMLSchema">
<xs:element name="audio" type="audio-type">

<xs:complexType name="audio-type">
 <xs:sequence>
    <xs:element name="frame" type="frame-type" minOccurs="1" maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>

<xs:complexType name="frame-type">
 <xs:sequence>
    <xs:element name="personID" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="time" type="interval-type" minOccurs="1" maxOccurs="1"/>
    <xs:element name="emotion" type=" emotion-type" minOccurs="1" maxOccurs="1"/>
 </xs:sequence>
 <xs:attribute name="frameID" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name=" interval-type">
 <xs:sequence>
    <xs:element name="start" type="xs:time" minOccurs="1" maxOccurs="1"/>
    <xs:element name="stop" type="xs:time" minOccurs="1" maxOccurs="1"/>
 </xs:sequence>
</xs:complexType>

<xs:complexType name="emotion-type">
 <xs:sequence>
    <xs:element name="happiness" type="xs:integer" minOccurs="1" maxOccurs="1"/>
    <xs:element name="sadness" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="fear" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="disgust" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="surprise" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="anger" type="xs:string" minOccurs="1" maxOccurs="1"/>
 </xs:sequence>
</xs:complexType>

</xs:schema>
```

Listing 14.3-6 *XML Schema for audio.speech.semantic.emotions.xml.*

```
<emotion>
{
        for $es in doc("audio.speech.semantic.emotions.xml")//audio/frame
        let $ev in doc("video.face.semantic.facialExpressions.xml")//video/frame [
        $es/personID=personID and xs:time($ev/time) >= xs:time($es/time/start) and  xs:time($ev/time)
        <= xs:time($es/time/stop) and
         xs:time($es/time/start) >= xs:time($time_start) and
        xs:time($es/time/stop) <= xs:time($time_stop)]
        order by $ev/time
        return
                <person personID="{$ev/personID}">
                        <video id={$ev/@frameID}>
                                $ev/time
                                $ev/emotion
                        </video>
                        <audio id={$es/@frameID}>
                                $es/time
                                $es/emotion
                        </audio >
                </person >
}
</emotion>
```

Listing 14.3-7 *Data preparation for semantic level data fusion for emotion recognition based on audio-video data.*

14.4 THE AUTOMATIC SURVEILLANCE APPLICATION

In this section we present a multimodal surveillance experiment based on the framework to illustrate the feasibility of our approach and to show how the framework can be used. The setting of the experiments is inside a Dutch international train (BeNeLux train) compartment. Data was captured of hired actors and train conductors, playing specific (normal as well as unusual) scenarios. The data was used as input to a surveillance application built on top of the framework.

14.4.1 Goal

Currently, BeNeLux train compartments are equipped with several microphones and cameras. Currently, the audio and video data captured by these sensors is transmitted to a central location, where operators have to monitor the data manually and take appropriate action when unusual events occur.

Figure 14.4.1 shows the interface an operator is confronted with. Our goal is to use the framework presented in this chapter to build a system to automate the manual inspection process currently performed by the human operators. More specifically, the system should detect unusual behavior in the train compartment and notify the operators. It is still the task of the operator to take the appropriate actions.

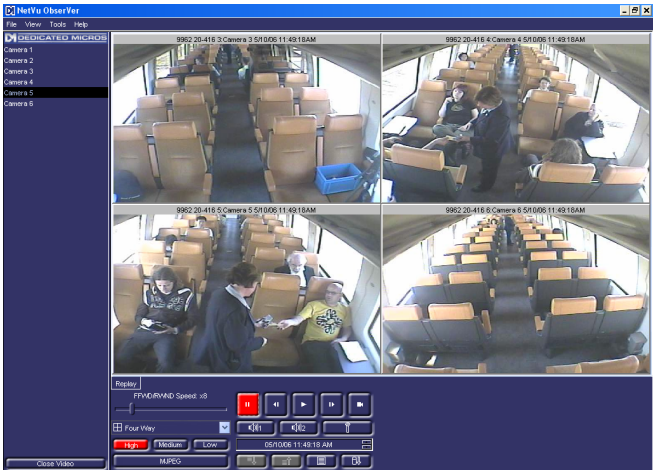

Figure 14.4.1 *User interface for train compartment surveillance operator.*

14.4.2 Experiment setup

In order to capture realistic data, professional actors and a train conductor were asked to play specific scenarios in the train compartment. We used four cameras and four microphones in the compartment to capture these scenarios.

As BeNeLux train compartments are already equipped with cameras, we have used these pre-installed cameras to capture video. The microphones however, where not located at the positions we preferred. So we installed four different microphones to capture audio data. As can be seen from Figure 14.4.2, the microphones do not cover the entire compartment. As a result, most of the unusual scenarios where played near the microphones.

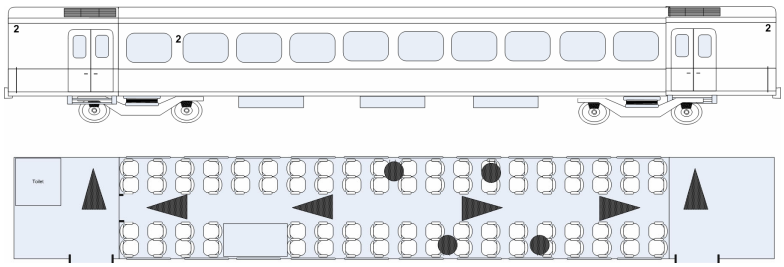

Figure 14.4.2 S*ide view of the train (top) and top view of the interior of a BeNeLux train compartment and the location of the sensors (bottom). The circles indicate the position of the omni directional microphones and the triangles indicate the cameras and their direction.*

The unusual scenarios we asked the actors to play fell in the category of the behaviors we want our system to detect, namely: fighting (including aggression towards the conductor and disturbance of peace), graffiti and vandalism, begging and sickness. The modules used from the framework to detect this behavior include face recognition component, gesture recognition component, face detection component, facial expression recognition component, and emotion recognition from speech component. The mapping between the behavior to recognize and the modules involved is given in Table 14.4-1.

Table 14.4-1
The mapping between the behavior to recognize and the framework modules involved.

Behavior to recognize	Services involved
Fighting	gesture recognition emotion recognition from speech
Graffiti and vandalism	Sound event recognition gesture recognition
Begging	gesture recognition motion tracking speech recognition
Sickness	gesture recognition

	emotion recognition from speech speech recognition

The resulting surveillance application that was built on top of the framework, consists of several interconnected detection modules made available by the framework. Each module is specialized in handling a specific tasks. For example, the sound event detection module detects whether there is someone shouting. The final module, called the aggression detection module, detects unusual behavior by fusing the results of different detection modules (Figure 14.4.3).

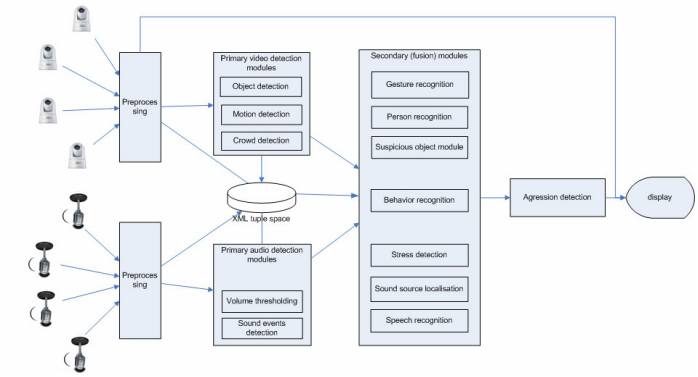

Figure 14.4.3 *Overview of the aggression detection system*

In the application, we have made a distinction between primary and secondary modules. Primary modules require real time data from the sensors and are continuously active. Primary modules are typically feature extraction (level 0) or object detection (level 1) algorithms. Secondary modules do not require real time attention and are typically triggered by the results of primary modules. For example, the object detection primary module fuses data from the cameras and detects a stationary object. This triggers the suspicious object secondary module that queries the objects history from the XML tuple space to determine the risk of the object.

14.5 CONCLUSION

The main contribution of this paper is two fold: it proposes a framework for automated multimodal surveillance designed to facilitate the communication between different processing components and second it presents the preliminary results of an ongoing surveillance project built on top of this framework. This framework is centered on the shared memory paradigm, the use of which allows for loosely coupled asynchronous communication between multiple processing components. This decoupling is realized both in time and space. The shared memory in the current design of the framework takes the form of XML data spaces. This suggests a more human-modeled alternative to store, retrieve and process the data. The data is conferred an underlying structure that complies with eXtended Markup Language (XML). Because the framework implementation itself relies on the philosophy of shared XML data spaces, special attention is given on how to integrate the two underlying technologies of processing components and XML data management.

In addition, the framework also consists of a set of software tools to monitor the state of registered processing components, to log different type of events and to debug the flow of data given any running application context. Depending on the type of application to be built on top of the multimodal framework, a routing algorithm has been design to manage the data transfer among existing shared memories on different physical networks.

Although the framework has been designed by taking into consideration the further development of an automatic surveillance oriented application, it can be adopted as basis for any kind of complex multimodal system involving many components and heavy data exchange. The specification fully complies with the requirements of data manipulation in a multi data producer/consumer context where the availability of data is time-dependent and some connections might be temporarily interrupted.

So far we have a prototype implementation for the presented multimodal workbench and our goal is to increase the number of processing components by implementing new algorithms or by modifying existing ones to fit into the framework.

References

[1] Cupillard, F., Avanzi, A., Bremond, F., and Thonnat, M. (2004). Video Understanding For Metro Surveillance. In Proceedings of the IEEE International Conference on Networking, Sensing & Control, Taipei, Taiwan.

[2] Velastin, S. A., et. al. (2002). A distributed surveillance system for improving security in public transport networks. Special Issue on Remote Surveillance Measurement and Control, 35(8):209-13.

[3] Kim, T. K.and Lee, S. U., Lee, H. J., Kee, S. C., and Kim, S. R. (2002). Integrated approach of multiple face detection for video surveillance. In Proceedings of the International Conference of Pattern Recognition (ICPR2002), page 394-397.

[4] Javed, O., Rasheed, Z., Alatas, O., and Shah, M. (2003). Knight, A Real-time Surveillance System for Multiple Overlapping and Nonoverlapping Cameras. In Proceedings of the International Conference on Multimedia and Expo (ICME 2003).

[5] Buxton, H. and Gong, S. (1995). Visual surveillance in a dynamic and uncertain world. Artificial Intelligence, 78(1-2):431-459.

[6] Foresti, G.L.; Micheloni, C.; Snidaro, L.; Remagnino, P.; Ellis, T., (2005), "Active video-based surveillance system: the low-level image and video processing techniques needed for implementation," Signal Processing Magazine, IEEE , vol.22, no.2 pp. 25- 37

[7] Shin, J. et. al., (2005), "Optical flow-based real-time object tracking using non-prior training active feature mode," ELSEVIER Real-Time Imaging, Vol. 11, pp. 204-218, USA.

[8] Zhu, Z, Li, W, Wolberg, G., (2005), "Integrating LDV Audio and IR Video for Remote Multimodal Surveillance," Computer Vision and Pattern Recognition, 2005 IEEE Computer Society Conference on , vol.3, no.pp. 10- 10, 20-26

[9] Talantzis, F., Aristodemos, P., Lazaros, P. C., (2006), "Real Time Audio-Visual Person Tracking", IEEE International Workshop on Multimedia Signal Processing

[10] Hongeng, S., Bremond, F., Nevatia, R., (2000), "Bayesian Framework for Video Surveillance Application," icpr, p. 1164, 15th International Conference on Pattern Recognition (ICPR'00) - Volume 1.

[11] Hai Tao; Sawhney, H.S.; Kumar, R., (2002), "Object tracking with Bayesian estimation of dynamic layer representations" in IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 24, Issue 1, pp: 75 - 89

[12] Pradeep, K. A.; Namunu, C. Maddage; Mohan, S. Kankanhalli, (2006), "Audio Based Event Detection for Multimedia Surveillance", in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICSSP06) - pp V813-816

[13] S. Spors, R. Rabenstein and N. Strobel, (2001), "Joint Audio-video Object Tracking", IEEE International Conference on Image Processing (ICIP), Thessaloniki, Greece.

[14] Zotkin, D. N., Duraiswami, R., Davis, L. S., (2002), "Joint Audio-Visual Tracking using Particle Filters" EURASIP Journal on Applied Signal Processing, Volume 2002, Issue 11, Pages 1154-1164

[15] Beal, M. J., Attias, H., Jojic, N., (2003), "A Graphical Model for Audiovisual Object Tracking", IEEE Transactions on Pattern Analysis and Machine Intelligence (Special section on graphical models in computer vision), pp 828-836

[16] Gatica-Perez, D. et. al., (2003), "Audio-visual speaker tracking with importance particle filters", in Proceedings of the International Conference on Image Processing (ICIP 2003) Volume 3, Page(s):III - 25-28

[17] Clavel, C., Ehrette, T. and Richard, G., (2005), "Events Detection for an Audio-Based Surveillance System", IEEE International Conference on Multimedia and Expo (ICME 2005), pp 1306- 1309

[18] Härmä, A., McKinney, M. F. and Skowronek, J, (2005), "Automatic surveillance of the acoustic activity in our living environment", In Proceedings of the IEEE Int. Conf. on Multimedia and Expo (ICME 2005)

[19] Goldhor, R. S. (1993), "Recognition of Environmental Sounds". In Proceedings of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP '93), volume 1, pp. 149-152

[20] Ivanov, Y., Stauffer, C., Bobick, A., (2001) "Video Surveillance of Interactions", IEEE Workshop on Visual Surveillance (ICCV 2001)

[21] Agrawal, R., et. al. (2001), "Vinci: A Service-oriented Architecture For Rapid Development of Web Applications", in Proceedings of the 10th International Conference on World Wide Web, pp 355-365.

[22] Hakeem, A., Shah, M., (2005), "Multiple Agent Event Detection and Representation in Videos", in Proceedings of the 20th National Conference on Artificial Intelligence, pp 89-94.

[23] Shankar R. Ponnekanti, Brad Johanson, Emre Kiciman and Armando Fox., "Portability, Extensibility and Robustness in iROS". Proc. IEEE International Conference on Pervasive Computing and Communications (Percom 2003), March 2003.

[24] Apparao, V., et. al., (1998), "Document Object Model (DOM) Level 1 Specification". W3C Recommendation http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001, World Wide Web Consortium.

[25] Clark, J. and S. DeRose, (1999), "XML Path Language (XPath)", Technical Report http://www.w3.org/TR/xpath, World Wide Web Consortium.

[26] Zhang, D. et. al., (2005), "Semi-Supervised Adapted HMMs for Unusual Event Detection", in Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1, pp: 611-618

[27] DiFranco, J. V., and Rubin, W. L., *Radar Detection,* Englewood Cliffs, NJ: Prentice-Hall, 1968.

[28] Blake, L. V., "Prediction of Radar Range*,"* Chapter 2 in *Radar Handbook*, M. I. Skolnik (ed.), New York: McGraw-Hill, 1970.

[29] Morchin, W., *Radar Engineer's Sourcebook,* Boston, MA: Artech House, 1993.

[30] Barton, D. K., and Ward, H. R., *Handbook of Radar Measurements,* Norwood, MA: Artech House, 1984.

[31] Bellur, U., Bodre, S.: 2006, 'xSpace – A Tuple Space for XML & its application in Orchestration of Web services'. pp. 766-772, SAC'06.

[32] Boag, S., Chamberlin, D., Fernández, M. F., Florescu, D., Robie, J., Siméon, J.: 2006, 'XQuery 1.0: An XML Query Language'. Candidate Recommendation http://www.w3.org/TR/xquery/, World Wide Web Consortium.

[33] Booth, D., C. K. Liu: 2006, 'Web Services Description Language (WSDL)'. Candidate Recommendation http://www.w3.org/TR/wsdl20-primer/, World Wide Web Consortium.

[34] Bray, T., J. Paoli, and C. M. Sperberg-McQueen: 1998, 'Extensible Markup Language'. Recommendation http://www.w3.org/TR/1998/REC-xml-19980210, World Wide Web Consortium.

[35] Fallside, D. C.: 2000, `XML Schema'. Technical Report http://www.w3.org/TR/xmlschema-0/, World Wide Web Consortium.

[36] Fontoura, M., Lehman, T., Nelson, D., Truong, T., Xiong, Y.: 2003, 'TSpaces Services Suite: Automating the Development and Management of Web Services'. ISBN 963-311-355-5.

[37] Thompson, P.: 2002, 'Ruple: an XML Space Implementation'. http://www.idealliance.org/papers/xmle02/dx_xmle02/papers/04-05-03/04-05-03.html

[38] Tolksdorf, R., F. Liebsch, D. M. Nguyen: 2004, 'XMLSpaces.NET: An Extensible Tuplespace as XML Middleware'. ISBN 80-903100-4-4.

[39] Zachariadis, S., L. Capra, C. Mascolo, and W. Emmerich: 2002, 'XMIDDLE: Information Sharing Middleware for a Mobile Environment'. In: Wireless Personal Communication Journal, ISSN: 0929-6212, Publisher: Springer Netherlands.

[40] Jini,. http://www.jini.org/

[41] Network Working Group RFC 3548: The Base16, Base32, and Base64 Data Encodings, 2003

[42] Khushraj D., Lassila, O., Finin, T., *sTuples: Semantic Tuple Spaces*, Proceedings of the Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04) 0-7695-2208-4/04, 2004.

[43] Datcu D., Rothkrantz L.J.M., "Automatic recognition of facial expressions using Bayesian Belief Networks", *Proceedings of IEEE SMC 2004*, ISBN 0-7803-8567-5, pp. 2209-2214, October 2004.

[44] D.Datcu, L.J.M.Rothkrantz, "FACIAL EXPRESSION RECOGNITION WITH RELEVANCE VECTOR MACHINES", *IEEE International Conference on Multimedia & Expo (ICME '05)*, ISBN 0-7803-9332-5, July 2005.

[45] W.S.Wong, W.Chan, D.Datcu, L.J.M. Rothkrantz, "Using a sparse learning relevance vector machine in facial expression recognition", *Euromedia'2006*, ISBN 90-77381-25-2, pp. 33-37, University of Ghent, April 2006.

[46] Datcu, D. and Rothkrantz, L.J.M., "The recognition of emotions from speech using GentleBoost Classifier", *CompSysTech'06*, June 2006.

[47] Datcu D., Rothkrantz L.J.M., "A multimodal workbench for automatic surveillance", *Proceedings of EUROMEDIA 2004*, ISBN 90-77381-08-2, pp. 108-112, April 2004.

[48] Rothkrantz L.J.M., van Vark R.J., Datcu, "Multi-medial Stress Assessment", *Proceedings of IEEE SMC 2004*, ISBN 0-7803-8567-5, pp. 3781-3786, October 2004.

[49] Xerces Parser, http://xml.apache.org/xerces-c/pdf.html