Adapting Indexation to the Content, Context and Queries Characteristics in Distributed Multimedia Systems

Mihaela Brut, Dana Codreanu, Ana-Maria Manzat, Florence Sèdes

2011 Seventh International Conference on Signal Image Technology & Internet-Based Systems

Adapting Indexation to the Content, Context and Queries Characteristics in Distributed Multimedia Systems

Mihaela Brut*, Dana Codreanu*, Ana-Maria Manzat* and Florence Sedes*
* Institut de Recherche en Informatique de Toulouse - Université Paul Sabatier
118 Route de Narbonne 31062 Toulouse Cedex 9, France
Email: {Mihaela.Brut, Dana.Codreanu, Ana-Maria.Manzat, Florence.Sedes}@irit.fr

***Abstract*—Due to the dramatically increasing amount of multimedia contents in several application domains, effective and flexible solutions for distributed data indexation are essential. In the management of these multimedia contents, the indexing process is the most important resource consumer, in terms of data transfer over the network and CPU consumption. As result of our research, we located two points for reducing resource consumption: the limitation of multimedia content transfer over the network for indexing, as well as the reduction of multimedia indexing amount by employing only the most appropriate algorithms for performing indexation, only over the relevant contents. We present in this paper the solution developed in the context of the LINDO project for indexing multimedia content into a distributed system that originally address these two points by reducing as much as possible the resource consumption through (1) a distributed indexing technique that avoids multimedia transfer, (2) a flexible mechanism for selecting the indexing algorithms to be employed on each remote server, according to the multimedia content characteristics, its acquisition context and user queries history.**

***Index Terms*—Multimedia Information Systems, Distributed Architecture, Dynamic indexation, Indexing algorithms selection**

I. Introduction

In the context of multimedia information management systems, contents are generally acquired and stored on different and heterogeneous locations, while their indexation is accomplished in real time or off-line, on the same server as their storage or on remote servers. The huge quantity of multimedia contents, the increasing number of remote servers data transmissions, as well as the management of the generated metadata constitute the main scalability issues that have to be handled by such systems. Two sensitive problems usually occur, which are discussed in this paper:

- *Architectural solutions*: most of the current systems define a specific architecture where the content is transferred to specialized indexation servers;
- *Indexation management techniques* employ usually a fixed predefined set of indexing algorithms.

The paper exposes a generic framework, developed in the context of the LINDO[1] (Large scale distributed INDexation of multimedia Objects) ITEA2 project, which is intended to guide the formalization and the development of a distributed multimedia information system, while favoring a reduce resource consumption, in terms of data transfers over the network, storage and CPU utilization. More precisely, the LINDO framework provides:

- an *architectural solution* applicable in many use cases (e.g. video-surveillance, broadcast and archive systems);
- a distributed and dynamic *indexing algorithms management*, enabling that any indexing algorithms could be integrated and deployed on demand on remote sites.

This paper proposes a technique for improving the dynamic indexing process by introducing new filtering criteria such as remote site characteristics and context, and also on the user queries. This technique reduces the indexing amount, and also dynamically establishes the most relevant indexing algorithms according to the changes that occur in the acquisition context (e.g. weather or luminosity variations).

In the remainder, the paper presents first how existing projects address the two mentioned issues. Then, the LINDO generic architecture is detailed in Section III. The indexation processes are detailed in Section IV through system workflows presentation (Section IV-A) and the description of the indexing algorithms selection mecanism (Section IV-B). In Section V, a solution for the optimization of the indexing process is proposed. Conclusions and further work directions are exposed in the end of the paper.

II. Related Work

Inside a distributed multimedia information system, the indexation process is managed through an indexation engine that includes multiple indexing algorithms. The indexation could be accomplished by applying on each multimedia content all these algorithms, or just a customized subset. In addition, a customized indexation could be realised by combining specific algorithms into some defined combinations. Also, the indexation could be accomplished in real time or off-line, at a central server level or distributed at the remote servers level. In the following, for some representative projects related to distributed multimedia information systems, we will briefly present: the main objective, the adopted architecture and indexation mechanism. This allows us to do some comparisons with LINDO approach in order to emphasize its advantages.

[1] http://www.lindo-itea.eu

DOI 10.1109/SITIS.2011.71

The CANDELA project[2] (Content Analysis and Network DELivery Architectures) proposes a generic distributed architecture for video content analysis and retrieval [1]. Multiple domain specific instantiations are realized (e.g., personal mobile multimedia management [2], video surveillance [3]). The indexation is uniformly accomplished in all remote servers, and managed at the central server level and the resulting metadata can be distributed over the network. However, the indexation algorithms are a priori selected and pre-installed.

Within its peer-to-peer architecture, the SAPIR project [4],[5] (Search on Audio-visual content using Peer-to-peer Information Retrieval) employs three specialized indexing servers (for images, texts and audio-visual contents) where each distributed peer sends its ingested content in order to be indexed. Both the multimedia content and the metadata issued from indexation are stored on each corresponding peer, while the user query is executed over the distributed peers.

The WebLab project[3] proposes an integration infrastructure that enables the management of indexation algorithms as web services in order to be used in the development of multimedia processing applications [6]. These indexing services are handled manually through a graphical interface. For each specific application a fixed set of indexing tools is run. The obtained metadata is stored in a centralized database.

The VITALAS project[4] (Video & image Indexing and retrieval in the Large Scale) capitalizes the WebLab infrastructure in a distributed multimedia environment [7]. The architecture enables the integration of partner's indexation modules as web services. The multimedia content is indexed off-line, at acquisition time, on different indexing servers. No selection of indexing algorithms based on user query is done.

The MUSCLE network of excellence[5] (Multimedia Understanding through Semantics, Computation and LEarning) develop a centralized framework, where the indexing algorithms are managed at the central server level, in order to be used for distributed indexation. Within the project a large set of different multimedia indexing algorithms has been developed (e.g. object recognition, content analysis, unusual behavior detection, movie summarization, human detection, speech recognition).

The KLIMT project (Knowledge InterMediation Technologies)[8] proposes a Service Oriented Architecture middleware for easy integration of heterogeneous content processing applications over a distributed network. The indexing algorithms are considered as web services. The user query is limited to pre-defined patterns that match a set of rules for the algorithms' execution sequence. After such a sequence selection, the content is analyzed and the metadata is stored in a centralized database.

In [9], a distributed image search engine based on mobile software agents is proposed. In order to deal with the metadata generated by the indexing algorithms, the authors propose two architectures: one centralizing the index and the other one distributing the indexes on the remote servers. The indexation is accomplished with a fixed set of indexing algorithms that are implemented as mobile agents. These agents migrate from one site to another in order to extract different multimedia features. Thus the multimedia contents are not transfered over the network.

[2]http://www.hitech-projects.com/euprojects/candela
[3]http://weblab-project.org/
[4]http://vitalas.ercim.org
[5]http://www.muscle-noe.org

TABLE I
A COMPARATIVE OVERVIEW OF SOME REPRESENTATIVE SYSTEMS AND APPROACHES

System	Architecture	Indexation
SAPIR	Peer-to-Peer	Uniform and fixed indexation at each pee
CANDELA	Generic distributed with central control	Uniform and fixed indexation at the same server as the content storage
VITALAS	Service oriented architecture	Variable set of indexing algorithms, running on some indexation servers
KLIMT	SOA with a central control	Realized at the query moment, with a variable set of IA, on dedicated servers
WebLab	SOA with central control	Realized at the acquisition moment, with a fixed set of IA, on dedicated servers
[9]	Distributed architecture based on mobile agents	Accomplished by a fixed set of mobile IA on the content server

An comparative overview of the most representative of these systems is presented in Table I.

After a comparative study on the concrete LINDO system needs, we decided to adopt a distributed architecture with a centralized management because it enables:

- To have a centralized management of indexing algorithms and also to deploy them on demand on different remote servers;
- To extract simultaneously multiple and diverse metadata by executing different indexation algorithms, in parallel, in different remote servers.
- To dispatch user queries only to some specific remote servers that might contain some desired information.
- To process queries simultaneously on the central server and on remote servers.

The indexation process adopted in the LINDO project is managed at the central server level and intelligently accomplished at the remote servers levels:

- A generic interface was defined for indexing algorithms in order to uniformly handle them [10], and to enable the integration of new algorithms at any time into the LINDO architecture;
- Two indexation processes were defined: (1) an implicit indexation that is a priori executed over the multimedia contents from each remote server; (2) an explicit indexation that could be executed, on demand, on a specific remote server.

III. THE LINDO GENERIC ARCHITECTURE

The main goal of LINDO project was to specify a generic architecture that could guide the design of distributed multimedia information systems, while favoring a reduce resource

consumption, in terms of data transfers over the network, storage and CPU consumption. Actually, the idea was not to define yet another information retrieval model or indexation engine, but rather to encapsulate any existing frameworks for indexing and retrieving multimedia contents, like those employed and/or defined by the projects presented in Section II.

Taking into consideration all these constraints, we have developed the LINDO generic architecture, illustrated in Figure 1, which enables each server to adopt uniform as well as differentiated indexations of multimedia contents. For that purpose, our architectural solution is divided into two main components: (1) *remote servers* which acquire, index and store multimedia contents, and (2) *a central server* which has a global view of the overall system and orchestrates the indexing and query processes.

In this architecture, each remote server is independent, i.e., it has its own specificities inside the distributed system, in terms of location, capacities, multimedia contents, indexation routines. For instance, some remote servers may index in real time acquired multimedia contents, while others may proceed to an off-line indexation. The central server has an important role in the architecture, as he can send relevant indexation routines or queries to relevant remote servers, while the system is running.

In the following, we will present and motivate each remote server module detailed in the lower part of Figure 1 (§III-A), as well as the central server ones (§III-B), detailed in the upper part of Figure 1. The interactions between the modules will be presented in Section IV-A.

A. The remote servers components

A remote server stores and indexes all acquired multimedia contents, and could provide as well answers to some queries. For that purpose several modules have been defined and composed together:

- *Storage Manager* (SM) stores the acquired multimedia contents, in real time or off-line. Through the *Transcode* module, a multimedia content could be converted into several formats, with different qualities and encodings,which allows an end-user to download different encodings of one desired content.
- *Access Manager* (AM) provides methods for accessing the multimedia contents stored into the SM. This module can also select different parts of one multimedia content. For instance, given two timestamps, it can select the corresponding clip from a video. This feature is useful when an end-user wants to see a specific event.
- *Feature Extractors Manager* (FEMrs) is in charge of managing and executing a set of indexing algorithms over the acquired multimedia contents. At any time, new algorithms can be uploaded into this module. Inversely, some algorithms can be updated or removed, if needed. It can permanently run some algorithms over all the acquired contents or it can execute them on demand only on certain multimedia contents. Another important functionality of this module is the automatic selection of the indexing algorithms that obtain the best performances in the current execution context (see Section IV-B).
- *Filtering module* compresses the outputs of an indexing algorithm that may contain redundant or useless metadata.
- *Metadata Engine* (MDErs) collects and aggregates all extracted metadata about multimedia contents. The metadata stored into this module can be queried in order to retrieve some desired information.
- *Time Client* is in charge with the time synchronization between the remote servers and the central server.
- *Service Description Controller* stores the description of the remote server, such as its location, its capacities, the software installed and the media acquisition context. In addition to this information, this module contains also the complete information about what indexing algorithms were executed and on which multimedia contents (see Section V-A).

B. The central server components

The central server has a global view of the whole distributed system. It can control the remote indexation processes, as well as answering or forwarding user queries to the remote servers that may contain pertinent results. One major difference between the central server and a remote server is that the central server does not store nor index multimedia contents. Actually, the central server can deploy, on the remote servers, indexing algorithms on demand. It is also in charge with the filtering of the multimedia content that has to be indexed. Thus, a central server is composed of the following components:

- *Terminal Interface* (TI) in which a user can specify some queries, in natural language or as keywords, and where the query results will be displayed. Inside the TI, several functions have been developed in order to visualize metadata collections, install, deploy and remotely execute some indexing algorithms.
- *Metadata Engine* (MDEcs) contains information related to the remote servers. It can contain some extracted metadata about distributed multimedia contents, some contextual information about the whole system, specific remote server acquisition contexts, the remote servers descriptions, such as their locations, their capacities, etc., but also some supplementary background knowledge.
- *Service Description Controller* collects all remote server descriptions, and aggregates them to the metadata collection in the MDEcs.
- *Feature Extractors Manager* (FEMcs) manages the entire set of indexing algorithms used in the system. It can deploy, update and remove any indexing algorithm, on any remote server, at any time. Furthermore, this module can execute and stop at any time an indexing algorithm.
- *Request Processor* (RP) processes the query in order to extract the demanded multimedia features, and executes these queries on the central server metadata engine (MDEcs) or forward queries on specific remote server metadata engines (MDErs). Based on the user query and on the information contained by the MDEcs, this module

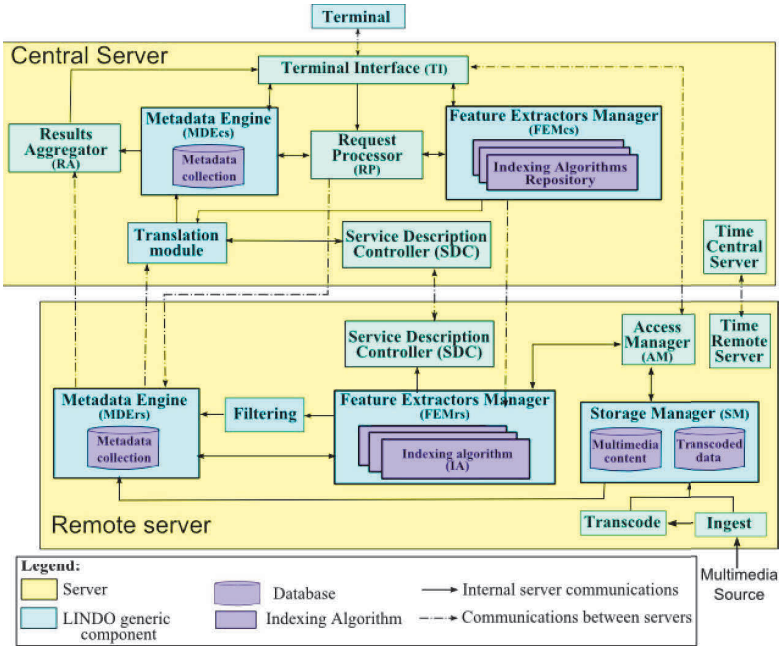

Fig. 1. The LINDO generic architecture.

decides if new indexation has to be accomplished on some remote servers, and, if it is the case, it selects the best indexing algorithms that should be employed.

- *Results Aggregator* (RA) aggregates the results received from the metadata engines. Actually, from a user query, it groups all the current available answers and sends them to the Terminal Interface module.
- *Translation module* homogenizes the data stored into the central server metadata engine. Indeed, many different models can be used by remote servers for storing the metadata, obtained after the multimedia contents indexation, or describing their characteristics. Hence, this module unifies all descriptions in order to provide one global view of the system.
- *Time Server* provides a unique system time that is used for synchronizing all remote server times.

In order to illustrate how the global system is running, we present in the next section two important system workflows related to multimedia contents indexation and retrieval.

IV. Implicit and Explicit Indexing Processes

A. General Presentation of System's Workflows

The above presented architecture permits the specification of several workflows that can be used for the implementation of different use cases. Moreover, in order to save server resource consumptions, multimedia content indexation is realized at ingest time (*implicit indexation*) and on demand (*explicit indexation*). Indeed, it avoids to execute all possible indexing algorithms at once. Figure 2 illustrates the indexing and querying workflows proposed by the LINDO framework.

When a new multimedia content is ingested by a remote server, its SM first stores it. Afterwards, the FEMrs module starts the *implicit indexation process (steps 2 to 19)*. This process consists in the execution of a predefined set of indexing algorithms, i.e., implicit indexing algorithms (iIA), on the acquired multimedia content in order to extract some metadata that will be further queried. For that purpose, the FEMrs retrieves the multimedia content from the SM through the AM module. Mainly in function of the media types and of the acquisition context, the FEMrs selects and executes the implicit indexing algorithms (steps 8 and 9). Once the execution of an indexing algorithm is achieved, the obtained metadata is forwarded to the Filtering module. The filtered metadata is then stored by the MDErs in its metadata collection. Each time the metadata collection is changing, the MDErs sends a concise version of the added metadata to the Translation Module on the central server[6]. Once the translation is done, the metadata are finally sent to the MDEcs in order to be stored and used for the querying process.

The query workflow begins with the user query specification by employing the TI. Each user query is sent to the RP module which analyses the query in order to translates it into the formal language used by the MDEcs and the MDErs. After this transformation, the RP selects, based on the information stored in the MDEcs, the remote servers that might have results to the query. The RP sends the query to the MDErs of the selected remote servers (steps 22 to 24), where the query is executed and the results are transfered to the RA module in order to be ranked and finally sent to the TI for displaying

[6][11] proposes an algorithm which computes several metadata summaries based on RDF descriptions.

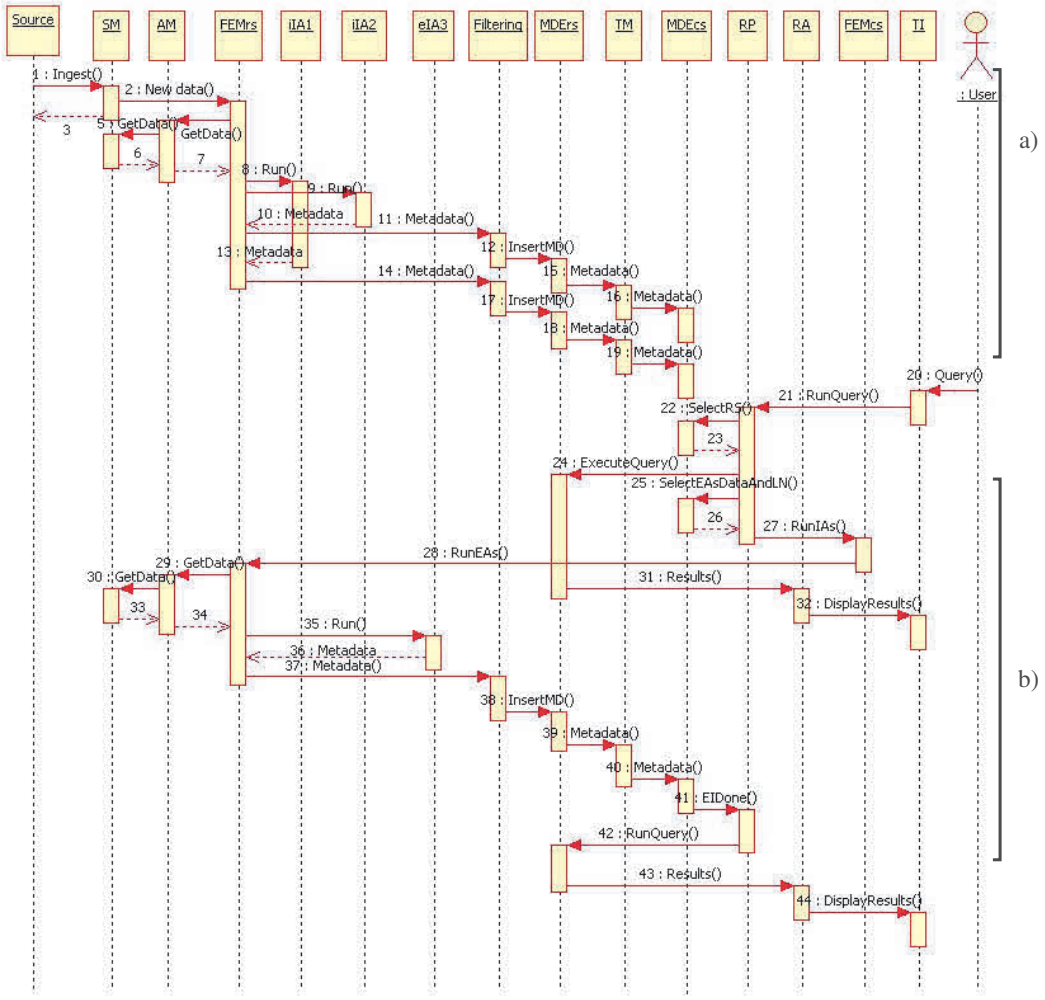

Fig. 2. Sequence diagram of a) implicit indexation; and b) explicit indexation.

(steps 31 and 32).

At this phase of the query process it is possible that not all the remote servers were selected for executing the query. This does not necessarily mean that they do not have results for the user query. It is possible that their multimedia content was not indexed with the right indexing algorithms. For this reason, the RP module selects the remote servers that should be re-indexed, the supplementary indexing algorithms, i.e., explicit indexing algorithms (eIA), that could be deployed on these remote servers in order to produce additional metadata that could provide answers to the user query (step 25). The user is informed that supplementary indexation is performed in order to retrieve other results to his query and that he will be announced when the results are available.

At this step, the RP decides, according to the user query, on what multimedia content these explicit indexing algorithms have to be executed. The FEMcs is in charge with the deployment of these explicit algorithms on the concerned remote servers, if necessary (step 28). Form this point the FEMrs takes the control of the indexation, which is accomplished as for the implicit indexation (steps 29,30, and 33 to 40). When the explicit indexation is finished, the MDEcs informs the RP that other results are available (step 41). At this moment, the RP module sends the user query to the MDErs of remote servers where the *explicit indexation* was accomplished. The results obtained are sent to the RA module which informs the user that supplementary results are available to his query (steps 43, 44).

In the next section we will detail the defined mechanisms for the selection of the indexing algorithms according to the user query.

B. The Selection Mechanism of Indexing Algorithms

As could be noticed, the explicit indexation process requires establishing a list of indexing algorithms based on the current user query: these algorithms are intended to produce metadata that could provide some relevant results for the given query. We developed a technique for indexing algorithms selection

[12] based on a uniform modeling of the query, the multimedia metadata and the indexing algorithm descriptions. This uniformity is acquired mainly by considering multimedia features in all models. More precisely, a query is viewed as a list of features to be retrieved, a multimedia metadata contains a list of features present in multimedia content and an indexing algorithm identifies a list of features.

Thus, given a user query Q that consist into a list of features $f_1, f_2, ..., f_n$ to be retrieved in the multimedia content, our technique generates, from the global algorithm list L_A, all the possible algorithms combinations L_{A1}, L_{A2}, ..., L_{Am} that could be executed in order to extract all the multimedia features $f_1, f_2, ..., f_n$. This technique consists in the following steps:

1) a feature f is selected from Q (among $f_1, f_2, ..., f_n$) according to the maximum number of indexing algorithms in L_A which extract f;
2) for each algorithm A that extract f, all the features from Q extracted by A are marked (including f);
3) for each un-marked feature from Q, a backtracking technique resume the Step 2 over the un-considered algorithms from L_A;
4) A solution is provided each time when all the features $f_1, f_2, ..., f_n$ became marked; the solution consists in a list of algorithms.

Each of the resulted algorithms combinations L_{A1}, L_{A2}, ..., L_{Am} produces therefore multimedia metadata including all the features $f_1, f_2, ..., f_n$ specified by the user query Q. In the next section we present a new mechanism, which dynamically selects among these variants the most suitable algorithm combination, by taking into consideration the characteristics of the multimedia collection: the selected algorithms should not only detect the requested multimedia features, but they should also have a good detection performance for the concrete multimedia contents acquisition context (weather, luminosity, location, speech language, content resolution etc.).

V. Dynamic Mechanism to update the Implicit and Explicit Indexation Algorithms

A. Semantic Descriptions of Remote Severs and of the indexing Algorithms

In our approach, a query is analyzed and processed in order to be associated with a list of multimedia features $f_1, f_2, ..., f_n$ that are of interest for the user. Based on this multimedia features list, the *Request Processor* selects first a set of remote servers where the query will be executed, as explained in §IV-B. This selection is based on the metadata contained by the MDEcs, received from the MDErs, and on the remote servers' descriptions.

For facilitating this selection, we included in the semantic description of each remote server some information about the algorithms that are already installed on the server (more precisely, the algorithms' names and descriptions). A matching will be accomplished between this information and the query.

In addition, the semantic description of a remote server includes information about the localization of the server, its domain of activity accompanied by an explanation, as well as the acquisition context for the managed multimedia content.

We consider bellow an example of semantic description for a server that concerns the video surveillance of a parking located in Paris where a person detection algorithm is installed. The server description includes also information about the evolution of the parameters that describe the weather and the luminosity conditions (expressed in %).

```
<RemoteServer id="rs1" name="RServer-1">
 <localisation>parking Austerlitz,
Paris, France </localisation>
 <description>Manages content from
cameras located in the parking in front
of the station </description>
 <devices>
   <camera id="c1Paris">
     <description>located in the north
corner</description>
   </camera>
 </devices>
 <acquisitionContext>
   <weather>
     <period start="2011-07-14T08:07:00"
end="2011-07-14T11:33:00">cloudy</period>
     <period start="2011-07-14T11:34:00"
end="2011-07-14T19:14:00">sunny</period>
   </weather>
   <luminosity>
     <period start="2011-07-14T08:07:00"
end="2011-07-14T11:33:00">75</period>
     <period start="2011-07-14T11:34:00"
end="2011-07-14T19:14:00">100</period>
   </luminosity>
 </acquisitionContext>
 <indexingAlgorithms>
   <indexingAlgorithm id="ia2rs1"
mediaType="video" name="person detection">
     <description>Detects persons
in outdoor area and their predominant
color</description>
   </indexingAlgorithm>
 </indexingAlgorithms>
</RemoteServer>
```

Moreover, the same list of multimedia features associated with a query is exploited in the case of explicit indexation in order to select the most suitable algorithms to be adopted in the indexation, according to the mechanism presented in the previous section. For this purpose, the semantic description of an algorithm includes information about the extracted multimedia features in the fields related to the algorithm name and to the description of its functionality.

In addition, in the semantic description of an algorithm we included information about the context of its execution, namely the conditions required for its best performance, such

as weather or luminosity conditions (in the next sub-section we present how these parameters are taken into account for increasing the efficiency of the indexing process).

The following XML fragment represents an example of such algorithm description.

```
<AlgorithmModel AlgoName="Person
Detection" MediaType="Video">
   <InputParameters>
      <InputParamFileFormat>xml
</InputParamFileFormat>
      <VideoParameters/>
   </InputParameters>
   <OutputObject Type="Metadata">
      <MetadataObject>
         <MetadataObjectDescription>
person and color detection
algorithm</MetadataObjectDescription>
      </MetadataObject>
   </OutputObject>
   <ExecutionConstraints>
      <MMConstraints>
         <weather>windy, cloudy</weather>
         <luminosity min="50" max="80" />
         <DataFormat>MPEG, AVI</DataFormat>
      </MMConstraints>
      <PlatformConstraints>
<OS>Windows</OS> </PlatformConstraints>
   </ExecutionConstraints>
</AlgorithmModel>
```

B. Considering Environmental Conditions (for Establishing Implicit/Explicit Algorithms)

As could be noticed, the semantic description of an algorithm provides information about the constraints that enable to obtain the optimal results further to the algorithm's execution. Such constraints include the quality of the multimedia content that is indexed, the weather conditions (sunny, cloudy, windy, rainy, snowy), the luminosity conditions (the degree of luminosity: night, day, murky, luminous, clear), location (indoor, outdoor), the language of the multimedia content (for speech detection), etc.

For the *implicit indexation*, a set of indexing algorithms is employed in order to obtain a specified set of multimedia features. For example, in a parking place, the implicit algorithms could concern person detection, car detection and registration plate detection. If for each of such detections, only one algorithm is employed, it is possible that the quality of detection would be affected by the changes encountered in the above mentioned environmental conditions.

For this reason, we propose for each multimedia feature that is intended to be detected implicitly on a remote server, multiple indexing algorithms to be included in the FEMrs, each one of them having different execution constraints. Based on this algorithms collection managed by FEMrs, we propose to dynamically change the implicit algorithms: when some environmental changes are encountered, the FEMrs will replace the current implicit indexing algorithm that detects a certain feature (e.g. human presence) with another algorithm that detects the same feature, but has a better performance in the current conditions.

For the parking example, we consider the weather and luminosity conditions. These changes are easy to capture based on some sensors, and are stored in the SDC module of each remote server (contributing to its description). Multiple "person detection" algorithms are available on this remote server, each of them having best performance in a certain weather conditions (sunny, cloudy, windy, rainy, snowy) and into a certain range of luminosity intensity (from 0% to 100%). Let us suppose that at the current moment the weather is sunny and on the remote server a "person detection" algorithm is running, with the best performance for "sunny" weather and "minimum 80% degree of luminosity". If a drastically weather change occurs, such as some dark clouds appear and it starts to rain, the sensors will transmit the modifications of the two considered parameters. Based on the semantic description of algorithms, FEMrs module will select another "person detection" algorithm to be executed, which has a good performance on a rainy weather, and in the conditions of a 40% luminosity degree.

As well, the environmental conditions are considered during the *explicit indexation*. Further to query analysis, there will be known the multimedia features that should be supplementary detected, as well as the multimedia sub-collection that should be indexed (e.g., corresponding to a certain time period, and to a subset of remote servers). Based on the mechanism presented in the sub-section IV-B, all the possible algorithms combinations are established that could extract the set of the multimedia features corresponding to this explicit indexation. Further, for each remote server selected for explicit indexation, the summary of its environmental parameters is consulted on the central server by the FEMcs. It splits the time period required by this indexation in some time sub-intervals, corresponding to changes encountered in the environmental parameters. For each sub-interval, FEMcs selects the algorithms combination that best suits the values of these parameters.The selected algorithms combinations are sent to the remote servers, where the explicit indexation is accomplished in a differentiate manner for each time sub-interval.

C. Considering the User Queries history

On each remote server, a fixed set of multimedia features is a priori established for being extracted during the implicit indexation (according to the characteristics of the remote server). Other multimedia features could be extracted normally during the explicit indexation, based on the user query.

We propose to consider the recurrent user queries during a time period in order to set up new implicit algorithms among the most demanded explicit algorithms. For this purpose, we associate a weight with each multimedia feature. It is possible that some features occur recurrently in the user queries during a certain time period. The weight of each

feature increases with each query that includes it. Our proposal consists in temporary considering a certain feature in the process of implicit indexation (more precisely, the algorithm that detects it) from the moment when its weight reaches a certain threshold. Of course, multiple algorithms could exist for detecting a certain multimedia feature, and the selection of the most suited one to the current context is accomplished as described in the previous sub-section. At the beginning of each time period (e.g., hour, day, week, etc.), the weight of all multimedia supplementary features are set up on zero, this avoiding that some temporary demanded features to be extracted permanently.

If we consider the parking example, it is possible that during a week multiple user queries concerning ”snatched bag” occur and determine the explicit execution on a certain remote server of the algorithm that detects ”snatched bag”. Consequently, the weight of the multimedia feature ”snatched bag” increases. When this weight reaches the considered threshold, the algorithm that detects ”snatched bag” becomes implicit. At the beginning of a new week, the weight is set on zero, while the algorithm still remains implicit. If during the current week, only few queries concern ”snatched bag”, the weight will remain low and the algorithm will not be kept among the implicit algorithms for the next week.

VI. Conclusions and perspectives

In the context of the LINDO project, this paper proposes a technique for improving the dynamic indexing process by introducing new filtering criteria such as remote site characteristics and context, as well as on the user queries. This technique reduces the indexing amount and dynamically establishes the most relevant indexing algorithms according to the changes that occur in the acquisition context (e.g. weather or luminosity variations). The LINDO framework was implemented and instantiated for video surveillance and broadcast use cases [13].

As future work, we will improve the indexing algorithm selection process by taking into account multiple characteristics from their semantic descriptions (e.g., their complexity, execution time). In the case of complex indexation needs, it is often necessary to execute into a certain order a chain of algorithms (e.g., French words detection algorithm must be executed only after a positive detection accomplished by the speech detection algorithm). Therefore, we intend also to include some pre-conditions and post-conditions specifications in the algorithms semantic descriptions and to exploit them in order to automatically determine such chains specific to a complex indexation need.

Currently, the relevance of a query result is established according to the precisions associated with the indexation algorithms used on each remote server. Hence, when a query is sent to multiple remote servers, the results provided by each of them are ordered according to these precisions. At the central server side, a general ranking of the overall results is accomplished. In the future, we plan to study different merging techniques for improving the global results relevance. More precisely, different result rankings will be compared and evaluated in order to establish their real impact to the global results list.

Acknowledgement

This work has been supported by the EUREKA project LINDO (ITEA2 – 06011).

References

[1] M. Petkovic and W. Jonker, *Content-Based Video Retrieval: A Database Perspective*, ser. Multimedia Systems and Applications. Berlin: Springer Verlag, 2003, vol. 25.

[2] P. Pietarila, U. Westermann, S. Jarvinen, J. Korva, J. Lahti, and H. Lothman, “Candela-storage, analysis, and retrieval of video content in distributed systems: Personal mobile multimedia management,” *Multimedia and Expo, IEEE International Conference on*, vol. 0, pp. 1557–1560, 2005.

[3] P. Merkus, X. Desurmont, E. Jaspers, R. Wijnhoven, O. Caignart, J.-F. Delaigle, and W. Favoreel, “Candela - integrated storage, analysis and distribution of video content for intelligent information systems,” in *European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology, EWIMT*, 2004.

[4] M. Agosti, E. D. Buccio, G. M. D. Nunzio, N. Ferro, M. Melucci, R. Miotto, and N. Orio, “Distributed information retrieval and automatic identification of music works in sapir.” in *SEBD*, M. Ceci, D. Malerba, and L. Tanca, Eds., 2007, pp. 479–482. [Online]. Available: http://dblp.uni-trier.de/db/conf/sebd/sebd2007.html#AgostiBNFMMO07

[5] B. Michal, F. Fabrizio, L. Claudio, N. David, P. Raffaele, R. Fausto, S. Jan, and Z. Pavel, “Building a web-scale image similarity search system,” *Multimedia Tools and Applications*, vol. 47, pp. 599–629, 2010, 10.1007/s11042-009-0339-z. [Online]. Available: http://dx.doi.org/10.1007/s11042-009-0339-z

[6] P. Giroux, S. Brunessaux, S. Brunessaux, J. Doucy, G. Dupont, B. Grilheres, Y. Mombrun, and A. Saval, “Weblab : An integration infrastructure to ease the development of multimedia processing applications,” in *the 21st Conference on Software and Systems Engineering and their Applications*, 2008.

[7] M.-L. Viaud, J. Thièvre, H. Goëau, A. Saulnier, and O. Buisson, “Interactive components for visual exploration of multimedia archives,” in *Proceedings of the 2008 international conference on Content-based image and video retrieval*, ser. CIVR ’08. New York, NY, USA: ACM, 2008, pp. 609–616. [Online]. Available: http://doi.acm.org/10.1145/1386352.1386440

[8] V. Conan, I. Ferran, P. Joly, and C. Vasserot, “KLIMT : Intermediations Technologies and Multimedia Indexing,” in *Third International Workshop on Content-Based Multimedia Indexing (CBMI’03), Rennes, France, 22/09/03-24/09/03*. INRIA, septembre 2003, pp. 11–18.

[9] V. Roth, J. Peters, and U. Pinsdorf, “A distributed content-based search engine based on mobile code and web service technology,” *Scalable Computing: Practice and Experience*, vol. 7, no. 4, pp. 101–117, 2006.

[10] M. Brut, F. Sèdes, and A.-M. Manzat, “A web services orchestration solution for semantic multimedia indexing and retrieval,” in *The 2nd Workshop on Frontiers in Complex, Intelligent and Software Intensive Systems*. IEEE Computer Society, 2009, pp. 1187–1192.

[11] S. Laborie, A.-M. Manzat, and F. Sèdes, “Managing and querying efficiently distributed semantic multimedia metadata collections,” *IEEE MultiMedia Special Issue on Multimedia-Metadata and Semantic Management*, vol. 16, no. 4, pp. 12–21, 2009.

[12] M. Brut, S. Laborie, A.-M. Manzat, and F. Sèdes, “A Framework for Automatizing and Optimizing the Selection of Indexing Algorithms,” in *the 3rd Conference on Metadata and Semantics Research*, pp. 48–59. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04590-5_5

[13] M. Brut, D. Codreanu, S. Dumitrescu, A.-M. Manzat, and F. Sedes, “A distributed architecture for flexible multimedia management and retrieval,” in *Database and Expert Systems Applications*, ser. Lecture Notes in Computer Science, A. Hameurlain, S. Liddle, K.-D. Schewe, and X. Zhou, Eds. Springer Berlin / Heidelberg, 2011, vol. 6861, pp. 249–263.