Content-Based Image Retrieval as a Web Service

Xavier Giro-i-Nieto, Carles Ventura, Jordi Pont-Tuset, Silvia Cortes, and Ferran Marques
Technical University of Catalonia (UPC), Barcelona, Catalonia / Spain
{xavier.giro, jordi.pont, ferran.marques}@upc.edu

ABSTRACT

This paper presents the design and implementation of a Content-Based Image Retrieval system where queries are formulated by visual examples through a graphical interface and content can be remotely accessed through web sevices. Visual descriptors and similarity measures implemented in this work are mostly defined in the MPEG-7 standard, while textual metadata is coded according to the Dublin Core specifications. Regarding the web service aspect, the presented solution is composed of two parts, a client running a graphical user interface to pose the queries and a server where the search engine explores an image repository. The separation of the user interface and the search engine follows a Service as a Software (SaaS) model, a type of cloud computing design where a single core system is online and available to authorized clients. The proposed architecture follows the REST software architecture and HTTP protocol for communications, two solutions that combined with metadata coded in RDF, make the proposed system ready for its integration in the semantic web.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: On-line Information Services—*Web-based services*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces

General Terms

Design,Management

Keywords

Content-Based Image Retrieval (CBIR), Web Service, Image Similarity, Information Visualization

1. MOTIVATION

The adage "A picture is worth a thousand words" has reached multimedia databases. This fact of experience has been heard true through ages, finding its origins in quotes attributed to the Chinese philosopher Confucius or the French general Napoleon. It took its definitive and worldwide known form as an advertisement from the American Fred R. Barnard in 1927. The slogan basically states that semantics can be more easily represented by images than by natural language. The advances in the digitization of visual data and their processing made possible implementing the concept behind the adage, boosting the development of techniques for their manipulation, making feasible the retrieval of images from databases based on their visual content.

The goal of a Content-Based Image Retrieval (CBIR) system [17, 18, 3] is to retrieve a ranked set of images from a target database. The resulting list is sorted according to the images similarity when compared to a user query. This similarity must be based on the visual features of the retrieved images, so one of the main challenges of such systems is the definition of a metric to assess this similarity. In addition, the success of a CBIR also depends on other characterists, such as its usuability and connectivity between the user interface and the target database. This paper presents the design and implementation of a CBIR system where queries are formulated by visual examples and content can be remotely accessed through web sevices.

CBIR systems typically support two types of queries: textual and visual. In the textual case, users normally formulate their queries using the classic natural language, the most common form of communication in human-to-human interaction. The main drawback of this technique is that images contain visual data that cannot be directly compared with text, so an intermediate process is necessary to assess similarity. The traditional option has been to ask a human expert to manually generate textual metadata for images in the target databases, a task that can be automatized in many cases thanks to appliying image processing and pattern recognition techniques A second group of queries require that users express their requests also visually, by providing examples, compositions, or sketches. This paper applies this second approach to the image retrieval problem, in the case where queries are formulated with examples.

Expressing queries through visual content may be, from a Human Computer Interaction point of view, a step backwards when compared to natural language solutions. The user cannot use speech nor writing to communicate with the system, the most common modalities for humans. Nevertheless, in many cases the visual mode becomes a simpler or the only way of formulation and search. These are the

situations where the user lacks words to express the semantics of the query or when the origin of the query is directly another image. A second advantage of visual queries is that it does not require any inter-modality conversion, as the visual descriptors of the query can be directly compared with the visual descriptors in the target database. Generating low level metadata is a matter of computational power and storage capabilities while textual metadata requires human interaction, a link in the chain that is more subjective, expensive and slower. For this reason, in this work image queries are processed according to the extracted visual descriptors. Four possible features are available for the user, together with their associated metrics that try to emulate the human visual perception.

If visual queries are valuable but they cannot be formulated using natural language, it is important to provide an interface as intuitive as possible. Graphical User Interfaces (GUIs) have been widely adopted to formulate visual queries applications because they share the same visual nature. Users understand GUIs as the access point to the database and they use them to both pose their queries and navigate through search results. These interfaces become the first layer between humans and the stored content.

Multimedia databases are evolving in parallel with another large-scale tendency in information technology. The popularization of the Internet has spread content in remote machines which are accessed through computer networks turning the world wide web into a world wide content repository. The physical distance between users and content has therefore grown, and the design of any information retrieval system should clearly differentiate between the user interface and the information service. The presented solution is composed of two parts, a client running the graphical user interface and a server where the search engine explores an image repository. The separation of user interface and the search engine follows a Service as a Software (SaaS) model, a type of cloud computing design where a single core system is online and available to authorized clients.

The presented work has been developed in collaboration with two industrial partners from the broadcast world. Their databases have been traditionally populated by manual metadata generated by professional archivers. For them, a CBIR system offers them an additional search criteria based on visual content of their videos. The described system is designed as a plug-in to their current Multimedia Asset Management (MAM) systems. This requirement has guided the design to create a clean separation between the CBIR server and any client that accesses to it. The proposed solutions have been inspired by the current web technologies and cloud computing principles.

This paper presents the elements and architecture of a query by example web service based on image global features. An introductory Section 2 provides a review of some of the CBIR systems proposed in the literature. Section 3 presents the search engine, visual descriptors, the similarity measures and fusion criterion applied. Section 4 introduces the graphical user interface and explains how the search engine parameters are collected and results displayed. Section 5 describes the system architecture and its dual desktop or web-service installations. Finally, section 6 draws the conclusions and points to future research directions.

2. RELATED WORK

Every CBIR system has a catalogue of visual features and similarity measures, which can be combined in multiple ways, always trying to fulfill the user expectations. Additionally, several approaches have been proposed in terms of user interfaces and architecture. This section provides an overview of related works that provided a complete implementation for CBIR.

CBIR systems appeared in the first half of the 90's. The QBIC system [4] developed by IBM was one of the first products and the similarity evaluation was based on color, texture, shape, and position. Queries could be based on the whole image or a part of it, sketches, and color and texture patterns. The system also included a strategy for high-dimensional feature indexing and its design was independent from the Graphical User Interface (GUI), allowing its integration in heterogeneous systems which facilitated its commercialization.

Later, systems like VisualSEEK [20] extended local queries at the region level to support the spatial relationships between the query parts. These systems require efficient indexing approaches for fast retrieval because the part-based nature of the query assessment rapidly boosts the amount of potential combination between query parts and target regions. VisualSEEK was developed following a server-client architecture on the WWW, where the client was a Java application and the server contained a test-bed of 12,000 images. The user interface also let the formulation of queries under the form of sketches by using a graphic editor. The server-side of the system was extended to WebSEEk [19], a web crawler that indexed distributed content on the WWW. The system retrieved images and metadata from remote repositories to extract their features and semantically map them to a set of predefined concepts. In this case, the user interface was directly coded in HTML to facilitate its access from a web browser.

Systems like MARS [14] or MIRROR [6] introduced humans in the retrieval loop to refine the user query. These systems asked users to flag which results matched their expectations and which did not. Their design was based on the assumption that the selection of the best visual features depends on both the data domain and user will at query time. By using this relevance feedback, the system can learn the selection of features and weights, and adapt the generated results to user expectation. MIRROR [6] supports query by example on the whole image based on one of the visual descriptors defined in the MPEG-7 standard, as well as suggested the similarity measures and evaluation dataset. The system offers two operational modes, an evaluation one that calculates the performance based on a ground truth, and another one that lets users choose what visual descriptor and similarity measure to consider. The interface to MARS is implemented in PHP to be accessed from a web browser. On the other hand, in the MARS system [14] the client was implemented as a Java applet, while the server part of the application was implemented in C++ and Java. The service was available online from a servlet running on an Apache Tomcat Servlet

container. The system could also be configured as a standalone application where client and search engine communicated directly with no need of an intermediate web server.

Apart from the relevance feedback contribution, the MARS project integrated techniques coming from the computer vision, database management systems and information retrieval. As a result, the system took special care of the Human-Computer Interaction (HCI), a research topic that had not been prioritized during the first decade of CBIR systems [23]. The visualization of results was targeted in different studies, focusing on the arrangement of results on the graphic panel [15], proposing alternatives to the classic grid distribution of result thumbnails [22, 24], or grouping similar results by applying clustering algorithms [21]. Another interesting topic was the collection of relevance feedback data in a simple and intuitive interface. ImageGrouper [11] substituted the MARS slider to quantize the user interest on every result for the creation of image groups. VisionGo [12] tried to cope with the variability of user expectactions by integrating in the GUI different relevance feedback strategies under the form of a recommender system. The response time was the key aspect aimed by CUZero [25], a search engine with a web interface that combines visual and semantic queries in a very short latency by using precomputed lists.

The presented work combines that most common features in a CBIR system and integrates them as a web service to be accessed from the industry partners. The state of the art REST software architecture is adopted together with a data repository compliant with the basic technologies proposed from the semantic web community. The design allows its integration on existing databases as external plug-ins that offer additional functionalities to the documentalists that manage and search in a broadcaster's video archive.

3. SEARCH ENGINE

The search engine implemented in this work is based on the computation of visual image descriptors and their similarity. Given a query image, a similarity measure between its descriptors and the rest of images' descriptors is calculated and the database is ranked in order of similarity with respect the query image.

The MPEG-7 [1, 2] standard proposes a set of visual descriptors, divided into color, shape, texture, etc., depending on the visual feature they refer to. The visual descriptors and similarity measures implemented in this work are mostly defined in this standard, together with some improvements, as presented below.

Following, the four visual descriptors implemented are briefly presented, along with the similarity measures used. Finally, we expound on how, given more than one visual descriptor, the results for each of them are fused together to rank the images by similarity.

3.1 Texture Edge Histogram

The *Texture Edge Histogram (TE)* represents the local spatial distribution of edges. A given image is first sub-divided into a grid of 4×4 subimages. For each of them, the response to a bank of five filters is calculated to extract the strongest direction, resulting in 16 histograms that add up to 80 bins.

To compare two Texture Edge Histogram descriptors, one global-edge (TE^g) and thirteen semiglobal-edge (TE^s) histograms are computed, by adding all (or some subsets of) the histograms of the 16 sub-images. The resulting MPEG-7 measure is defined as:

$$D(TE_1, TE_2) = \sum_{i=1}^{80} |TE_1(i) - TE_2(i)| + 5 \sum_{i=1}^{5} |TE_1^g(i) - TE_2^g(i)| + \sum_{i=1}^{65} |TE_1^s(i) - TE_2^s(i)|$$

where $TE(i)$ is the ith bin of the histogram.

3.2 Color Structure

The *Color Structure (CS)* computes the color histogram taking into account the local spatial structure. A structuring element sweeps the whole image, and at each position the bins of the colors that appear are incremented in one unit, independently of the number of pixels of that color within the structuring element.

The distance proposed in MPEG-7 for this descriptor is the L_1-norm [10] between the two histograms CS_1 and CS_2:

$$D(CS_1, CS_2) = \sum_{m=1}^{M} |CS_1(m) - CS_2(m)|$$

where M is the number of bins and $CS(m)$ refers to the mth bin of the histogram.

3.3 Color Layout

The *Color Layout (CL)* captures the global spatial layout of the colors in an image by computing the DCT coefficients of the set of mean colors of a 8×8 grid. Mathematically, the descriptor is represented by N_i coefficients for each of the N_c channels, as follows:

$$CL = \left\{ \left\{ c_j^i \right\}_{j=1}^{N_i} \right\}_{i=1}^{N_C}$$

Two descriptors CL_1 and CL_2 are compared in MPEG-7 as follows:

$$D(CL_1, CL_2) = \sum_{i}^{N_c} \sqrt{\sum_{j}^{N_i} w_j^i \left(c_{j\,1}^i - c_{j\,2}^i \right)^2}$$

where w_j^i is the weight associated to the j-th coefficient of the i-th channel, allowing the measure to penalize more the differences in lower frequencies.

3.4 Dominant Color

The *Dominant Color (DC)* describes an image by a small number N of representative color values $\mathbf{c}_i$, their percentages p_i and variance v_i, along with the mean spatial coherency s:

$$DC = \{(\mathbf{c}_i, p_i, v_i), s\}, \qquad (i = 1, 2, ..., N) \qquad (1)$$

Given two descriptors DC_1 and DC_2, each pair of colors $\mathbf{c}_{1i}$ and $\mathbf{c}_{2j}$ are considered as similar if their Euclidean distance

$d_{1i,2j}$ is below a threshold T_d. A *similarity coefficient* $a_{1i,2j}$ is then defined as:

$$a_{1i,2j} = \begin{cases} 1 - d_{1i,2j}/T_d & d_{1i,2j} \leq T_d \\ 0 & d_{1i,2j} > T_d \end{cases}$$

The value of T_d was manually set to 16, as proposed in [1]. Given these coefficients, MPEG-7 proposes the following dissimilarity measure:

$$D^2(DC_1,DC_2) = \sum_{i=1}^{N_1} {p_{1i}}^2 + \sum_{j=1}^{N_2} {p_{2j}}^2 - \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} 2\, a_{1i,2j}\, p_{1i}\, p_{2j} \quad (2)$$

One of the problems detected about this measure arises when comparing the query image and a target image whose pairs of matched dominant colors are not close enough. In these cases, the parameters $a_{1i,2j}$ from the dissimilarity measure in Equation 2 are almost 0, and therefore the the third summation in the equation becomes negligible. Then, the target images with lowest distance will be the ones with p_{2j} more homogeneously distributed, because the second summation of the equation is low in these cases. This behavior is not perceptually meaningful.

In order to tackle this problem, a variant of this distance has been implemented. It is based on ensuring that a minimum percentage of pixels of the pair of images (query and target) belongs to the dominant colors that have been matched. In other words, a minimum portion of the image has to be represented by the dominant colors that are less than T_d distance apart. This way, the larger the minimum percentage, the more accurate the descriptor.

The *Merged Palette Histogram Similarity Measure* (MPHSM) [13] has also been experimented for comparing Dominant Color descriptors. It is based on generating a common palette for the two dominant color descriptors by merging their color histogram bins. This is done in order to use the conventional histogram intersection similarity measure [16]. In [13], the common palette is generated by iteratively searching the closest two colors between the palettes. If this minimum distance is lower than the threshold T_d, the two colors are merged as:

$$c_{m(i,j)} = \frac{p_{1i}c_{1i} + p_{2j}c_{2j}}{p_{1i} + p_{2j}} \quad (3)$$

This process continues until the minimum distance is greater than T_d. The main drawback about this process is that it leads to a matching that potentially is not the one that minimizes the global distance between the matched pairs of colors.

In order to obtain a global optimum, the following variation is included. Given two palettes:

$$P_1 = \{\mathbf{c}_{11}, ..., \mathbf{c}_{1N_1}\} \quad \text{and} \quad P_2 = \{\mathbf{c}_{21}, ..., \mathbf{c}_{2N_2}\},$$

a bipartite graph that contains an edge between any pair of colors $\mathbf{c}_{1i}$ and $\mathbf{c}_{2j}$ is defined. The weight of the edges is the distance between the paired colors. Then, the objective is to find the minimum matching between the colors of P_1 and P_2, i.e., the one with lowest weight.

This problem is commonly known as *the assignment problem* and can be efficiently solved using the Hungarian algorithm [7]. Thus, the colors from palette P_1 that had been matched to colors from palette P_2 by the Hungarian algorithm are merged as in Equation 3. The common palette:

$$\left\{\{\mathbf{c}_{mi}\}, i = 1, ..., N_m\right\}$$

with N_m colors ($N_m \leq N_1 + N_2$) is then generated with the use of the merged and non-merged colors from the two palettes.

This merged palette forms a common color space for the two histograms, so each dominant color descriptor, DC_1 and DC_2, can be redefined in this space. As these two histograms, DC_{1m} and DC_{2m}, are based on a common color palette, the histogram intersection method can be directly applied in order to compute their similarity. The MPHSM value is defined as the intersection area of these two histograms, which is given by [16]:

$$I(DC_{1m}, DC_{2m}) = \sum_{i=1}^{N_m} min(p_{1mi}, p_{2mi})$$

The larger the value $I(DC_{1m}, DC_{2m})$, the more similar the two images, being 1 the maximum value.

3.5 Fusion by Ranking

Given the query image and a computed visual descriptor, the search engine computes the similarity between this visual descriptor and the descriptor of each image in the target database, providing a ranking of these images sorted in order of similarity. Formally, we denote the position of the target image T_i in the ranking as $p_i \in \mathbb{N}$. The most similar image, for instance, will have $p_i = 1$.

Having a set of N_d descriptors and the ranking of each target image for each of them, the dissimilarity gathering all the descriptors is defined as:

$$D(Q, T_i) = \sum_{k=1}^{N_d} w_k\, p_{ik}$$

where p_{ik} is the position of the target image T_i in the rank corresponding to the k-th descriptor and w_k is its associated weight. These weights allow the user to give more importance to some descriptors, making the application more flexible.

Sorting all the target images in the database in order of $D(Q, T_i)$, the search engine provides a global ranking gathering the results from all the descriptors.

4. USER INTERFACE

The user interaction with the query by example system is managed by a graphical interface named GOS (Graphical Object Searcher). The first task of this tool is the translation of a human query into a set of configuration parameters required by the search engine. Afterwards, when the results are available, the same interface shows the retrieved content in a framework for navigation and browsing.

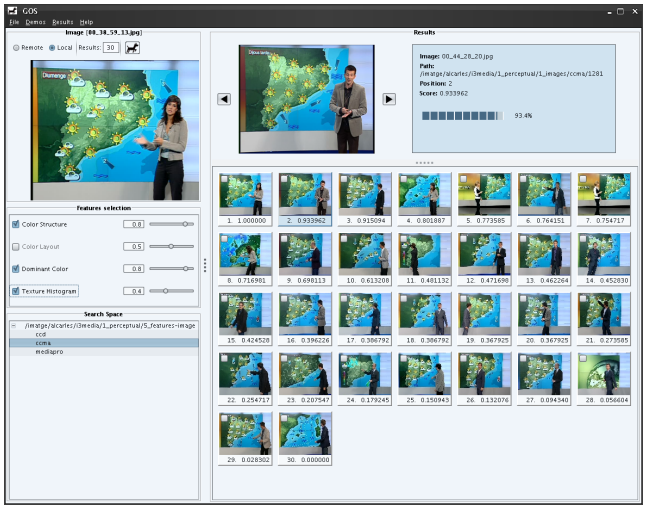

Figure 1: GOS Graphical User Interface.

The design of the interface follows User-Centered Design (UCD) because the final goal of the system is to provide users with results that satisfy their expectations. The Human-Computer Interaction principles state that the way of showing information to users affects their behavior, so will have an impact on the usability of the system and the final satisfaction degree.

Figure 1 shows the GOS interface after a search process. The distribution of the panels follows the reading pattern of the target users: from left to right, and from the up to down. The application window is vertically divided in two main areas, the one on the left grouping the query parameters and the one on the right showing the retrieved images

The proposed layout follows the natural sequence of the user actions: firstly the query formulation and afterwards the results browsing. The focus starts on the upper-left part of the interface, where the user can always see the query image. This upper part of the query panel also includes a cell where the amount of results can be set and two radio buttons where users are prompted for a local or remote execution of the search engine, as it will be later explained in Section 5. Once the query image is selected, the panel below shows the list of available visual descriptors with checkboxes next to them that let users select which are to be considered for the current query. Next to the selected descriptor's name, a slider allows setting the weights that express the relevance of each descriptor in the query. Finally, the last panel in the query area shows a tree whose nodes represent image collections in the target database. Users can reduce the search space by selecting those nodes that interest them and, by doing so, obtaining faster results on a more precise set.

The pointing dog icon on the upper right corner in the query panel launches the search engine, which returns to the interface a rank list of the top images in the search space and their associated distances. Results are shown in the central/right part of the window, an area which is vertically divided again to show in its lower part the thumbnails and scores of the retrieved images, and in its upper part a detailed visualization of one of the results. Shown thumbnails are precomputed in the target database for a faster response and are sorted in the grid according to the rank provided by the search engine. Together with their position in the rank list, a label below every thumbnail shows its score, obtained by normalizing the associated distance by the highest retrieved distance. Users can focus on a thumbnail by clicking on it or by using the mouse wheel, that controls a highlight frame running through the thumbnails. When a result is highlighted, the upper section of the results panel shows a higher definition version of the retrieved image together with the associated textual metadata. A checkbox is inserted on each thumbnail so that users can select them and save the list in an MPEG-7/XML compliant file.

The usability of the GUI is completed by a configuration file that stores the default values of some parameters and data paths, as well as all the required information to communicate with the search engine. The tool has been implemented to support multi-lingual messages, an effort to increase usability by adapting the interface to the users language. Finally, GOS also includes help documentation with step by step instructions for all basic operations.

5. ARCHITECTURE

After the presentation of the search engine and the user interface in Sections 3 and 4, this part describes the proposed architecture for the deployment of the described system. Two basic solutions have been implemented according to the users requirements. The first set of users are the researchers who develop the search engine. These users usually work on a fixed dataset located in their local filesystem and are interested in the internal execution and steps of the process. On the other hand, the documentalists at the broadcaster's archive want to perform searches on the company's repository. These users understand CBIR as an add-on to the existing text-based queries. In the first case, the presented tools are executed locally as a desktop application while in the second case, a remote execution of the search engine is controlled from the local GUI.

Although the final goal of the system is video retrieval, the chosen approach is based on keyframes. Both companies own keyframe extractors that process their video assets before entering the CBIR system. The use of keyframes significaly decreases the amount of data to process and, if the keyframe extractor provides enough representative frames from a semantic point of view, this simplification is worthy from the final application point of view. A good keyframe extractor should minimize the amount of similar keyframes in the same video asset so that a query by example process would not retrieve many images from the same video asset.

Figure 2 shows the architecture elements and their interconnections. Users formulate queries and view results on the GUI. In the local case, the GUI writes to disk a configuration file and invokes the search engine with an operative system call. In the remote case, an HTTP message is sent through an IP network to an applications server, which passes the request to a servlet that manages the execution of the search engine on another servlet that manages the target repository.

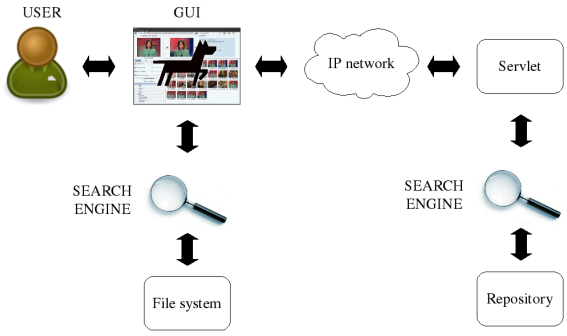

Figure 2: System architecture.

5.1 Desktop application

In the local execution mode, the GUI is configured with the path to the search engine and paths to temporary directories where log files are generated. Target databases are defined by disk directories that contain a file for each image in the database. This file is an MPEG-7/XML document that contains the visual descriptors extracted from the image as well as the path of the original image and its thumbnail. All these data have been previously generated and are shared by all researchers.

The search process starts by choosing one descriptors file in the target database, one or several directories as search spaces and the desired weights for considered descriptors. The GUI scans the selected directories to build a search space file that is provided to the engine, together with the query image and configuration parameters. When results are ready, they are written to disk and the GUI reads them to display the thumbnails of the top ranked images.

5.2 Web services

In the web service mode, the basic scheme considers one or several instances of the GUI running in user machines that connect to a remote server where the search engine is installed. Together with the engine, this server also hosts the target database of the queries by example.

5.2.1 Repository

The image database is implemented by an installation of the Fedora Commons Repository [8], a tool that manages content and metadata. This software was a requirement from the broadcast companies who wanted a technology compliant with semantic web technology for the integration with third-party applications. The Fedora data model is based on the concept of *digital objects*, and each digital object has associated a unique Persisten ID (PID) and datastreams of content and metadata. There are two types of metadata: Dublin Core and semantic. Dublin Core (DC) is a recognized standard for metadata and is composed by fifteen elements that can naturally code the type of textual metadata used in the broadcaster's archives. Attributes such as the *DC:title*, *DC:description* or *DC:date* of the video assets are defined by the standard and supported by the Fedora Commons Repository. These metadata are indexed in a SQL database that can process the most common queries by text. Additionally, Fedora Commons supports the definition of additional semantic information in the Resource Description Framework

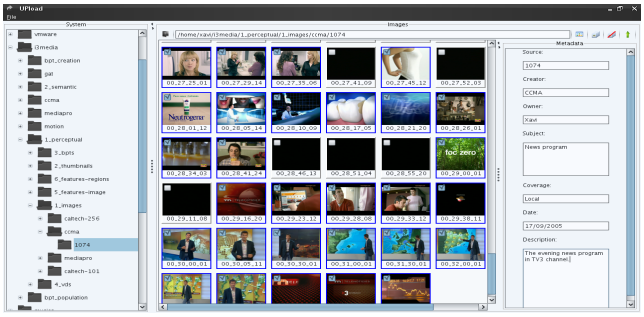

Figure 3: Screenshot of the ingest interface.

(RDF) promoted by the W3C. This second type of metadata is managed by another database, an installation of the Mulgara semantic database.

In our system, two data models have been defined for digital objects: the *image* and *video* models. Image objects have datastreams attached that contain the ingested data coded in JPEG, a thumbnail, and the MPEG-7 visual descriptors described in Section 3. On the other hand, video objects represent collections of image objects. The video objects are semantically related to their keyframes, and vice versa, a semantic information that is kept in the semantic database. Additionally, both image and video objects have Dublin Core metadata that is manually entered to the system during content ingest. A second GUI, called *UPLoad*, has been developed to easily add keyframes and metadata to the repository. Figure 3 includes a screenshoot of this tool in the case that the keyframes from a TV news program are ingested to the repository. In this example, all image objects will be linked to a video object labelled *1074* in the *DC:source* field, and whose *DC:creator* value is *CCMA*, the broadcaster that produced the boulletin.

Whenever a new image is ingested to the system, an image digital object is created and a unique identifier is assigned to it. Afterwards, a datastream is attached to it with the image content and so is a second datastream with the available Dublin Core metadata. In parallel, a messaging service implemented in Fedora Commons is activated to process the image asynchronously, without blocking the ingest of new images. For each image, a thumbnail is generated and its visual descriptors extracted and attached as datastreams to the digital object.

The combination of Dublin Core and RDF metadata is used to dynamically propose search spaces to users. Each digital object in the system includes among its Dublin Core metadata the *DC:creator*. This field describes the person or entity responsible for making the resource. The system is designed for users that work on content produced by a single creator, a value specified in the configuration file of the GUI. The search spaces shown in the GOS panel are defined by retrieving those video objects that match the requested creator. When users select a set of the available search spaces, the target dataset is build by following the RDF relations between the video objects and the associated image objects.

5.2.2 Servlet and REST design

Another element in the system, a Java servlet, is responsible for the coordination between the search engine and the repository in the server. This application runs online on an Apache Tomcat applications server and awaits from incoming requests from the client GUIs. When a query is received, it collects the search spaces chosen by the user to build the target dataset and executes the search engine with the selected descriptors and weights; the same way that the GUI runs in local execution mode. When results are available, they are sent back to the GUI to be shown to users on the results panel.

The servlet is also responsible for the HTTP communication between the web service and the client. All configuration parameters are directly coded in an HTTP GET method that requests the representation of a web resource. The client waits until the execution of the search finishes and the results document is returned. The MPEG-7/XML document coded the PIDs of the retrieved digital objects in the repository as well as their score and position in the rank list. With this information, the client GUI accesses the repository to retrieve the thumbnails and image datastreams required to refresh the display. Additionally, the PIDs are also used to retrieved the Dublin Core metadata shown in the results panel.

Figure 4 shows the complete diagram of the communication between the system elements. The first message is sent when the GUI retrieves from the repository the set of video PIDs (vPID) associated to the desired creator. The query by example process starts with the creation of an image object for the query, which is assigned the image identifier $iPID^Q$. With this unique value, the GUI ingests the image in the repository and its visual descriptors are computed in the server. The search process is started by another message that codes the query in terms of $iPID^Q$ and the selected set of $vPID^{SS}$ that define the search space selected by the user. At this wait, the GUI waits for the servlet to retrieve the set of images $iPID^{SS}$ that define the target database and run the engine on them. When the ranked set of results images $iPID^R$ is sent back to the GUI, it retrieves the URLs of the associated thumbnails and, in the case of the result in focus, also is full definition version and Dublin Core metadata.

The described architecture follows the principle of a REpresentation State Transfer (REST) design, a common practise in Internet web services. The concept considers a client-server model that communicates by transfering *representations* of *resources*. In the presented CBIR, the resource is the *similar images* service that takes different representations in the form of MPEG-7/XML document, depending on the input query by example parameters. The client GUI can be in two basic states from the server point of view, whether in *rest* while interacting with the user, or transitioning between query collection to results display while waiting for a response. The server is stateless, as it does not keep any record from the client state as every request is self-contained, a property that greatly simplifies the design. The implementation uses the Java classes provided by the Restlet project [9], an API that manages the communications between client and servlet following the REST principles.

6. CONCLUSIONS

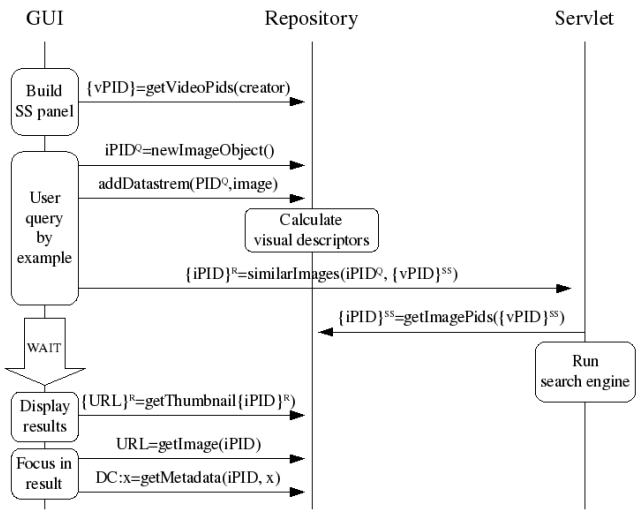

Figure 4: Communication diagram.

This paper has offered a detailed description of the system architecture of a Content-Based Image Retrieval designed as a web service. Each of the composing parts has been presented and the involved technologies discussed. The final result is a modular system where most of the processing effort can be located on the server while the client is designed to collect the user requests in a usable and intuitive environment. The optoin of local execution is provided so that the GUI can also be used during the development of the search engine.

The separation between client and server opens the door to other implementations for clients. The API between the two sides of the system is clearly specified thanks to the adopted REST design. This architecture simplifies the access to the service from any type of client device. In fact, the presented service is already accessed by a third-party industrial application implemented on Adobe Flex [5]. Additionally to desktop applications, the CBIR service could be accessed from mobile terminal or TV set top boxes, that would need to adapt the graphical interface to the device and user interaction, but that could use the same HTTP methods.

The use of standardized technology has been one priority in this project. Visual descriptors and query by example results are coded in MPEG-7/XML, a reference standard for multimedia content management. The metadata indexed in the Fedora Commons repository contains higher level concepts according to the Dublin Core field and the RDF semantic language. In the communication layer, client requests and server responses are implemented in the popular HTTP protocol. The adoption of standardized formats facilitates the connectivity of the web service to any third-party client, as well as any future exportation of content or metadata.

The presented project has settled the framework for the future development of new features that will improve the CBIR user experience. Planned enhancement focus on two main aspects: a faster response by using efficient index strate-

gies, and more satisfying results by introducing relevance feedback techniques.

7. ACKNOWLEDGMENTS

All images used in this paper belong to TVC, Televisio de Catalunya, and are copyright protected. They have been provided by TVC with the only goal of research under the framework of the i3media project. This work was partially founded by the Catalan Broadcasting Corporation (CCMA) and Mediapro through the Spanish project CENIT-2007-1012 i3media, and by TEC2007-66858/TCM PROVEC project of the Spanish Government.

8. REFERENCES

[1] I. 15938-3:2001. Multimedia content description interface - part3: Visual. Version 1.

[2] P. B. S. Manjunath and T. Sikora. *Introduction to MPEG-7, Multimedia Content Description Interface*. John Wiley and Sons, Ltd., Jun 2002.

[3] R. Datta, J. Li, and J. Z. Wang. Content-based image retrieval: approaches and trends of the new age. In *MIR '05: Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, pages 253–262, New York, NY, USA, 2005. ACM.

[4] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: the qbic system. *Computer*, 28(9):23–32, Sep 1995.

[5] X. Giro-i Nieto, R. Salla, and X. Vives. Digimatge, a rich internet application for video retrieval from a multimedia asset management system. In *ACM SIGMM International Conference on Multimedia Information Retrieval*, Mar. 2010.

[6] W. K, C. K, and P. L. Mirror: an interactive content based image retrieval system. In *ISCAS '05: Proceedings of the IEEE Intl. Symposium on Circuits and Systems*, 2005.

[7] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955.

[8] C. Lagoze, S. Payette, E. Shin, and C. Wilper. Fedora: an architecture for complex objects and their relationships. *International Journal on Digital Libraries*, 6(2):124–138, April 2006.

[9] J. Louvel and T. Boileau. *Restlet in Action*. MEAP Began, Nov 2009.

[10] D. Messing, P. van Beek, and J. Errico. The mpeg-7 colour structure descriptor: image description using colour and local spatial information. In *International Conference on Image Processing*, volume 1, pages 670–673, 2001.

[11] M. Nakazato, L. Manola, and T. S. Huang. Imagegrouper: a group-oriented user interface for content-based image retrieval and digital image arrangement. *Journal of Visual Languages and Computing*, 14(4):363 – 386, 2003. Biomedical Visualization for Bioinformatics.

[12] S.-Y. Neo, H. Luan, Y. Zheng, H.-K. Goh, and T.-S. Chua. Visiongo: bridging users and multimedia video retrieval. In *Proceedings of the 2008 international conference on Content-based image and video retrieval*, pages 559–560, New York, NY, USA, 2008. ACM.

[13] L.-M. Po and K.-M. Wong. A new palette histogram similarity measure for mpeg-7 dominant color descriptor. In *International Conference on Image Processing*, volume 3, pages 1533–1536 Vol. 3, Oct. 2004.

[14] K. Porkaew and K. Chakrabarti. Query refinement for multimedia similarity retrieval in mars. In *MULTIMEDIA '99: Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 235–238, New York, NY, USA, 1999. ACM.

[15] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. Does organisation by similarity assist image browsing? In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 190–197, New York, NY, USA, 2001. ACM.

[16] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, Nov 2000.

[17] Y. Rui, T. S. Huang, and S.-F. Chang. Image retrieval: Current techniques, promising directions, and open issues. *Journal of Visual Communication and Image Representation*, 10(1):39 – 62, 1999.

[18] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, 2000.

[19] J. Smith and S.-F. Chang. Visually searching the web for content. *Multimedia, IEEE*, 4(3):12–20, Jul-Sep 1997.

[20] J. R. Smith and S.-F. Chang. Visualseek: a fully automated content-based image query system. In *MULTIMEDIA '96: Proceedings of the fourth ACM international conference on Multimedia*, pages 87–98, New York, NY, USA, 1996. ACM.

[21] D. Stan and I. K. Sethi. eid: a system for exploration of image databases. *Inf. Process. Manage.*, 39:335–361, May 2003.

[22] R. S. Torres, C. G. Silva, C. B. Medeiros, and H. V. Rocha. Visual structures for image browsing. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 49–55, New York, NY, USA, 2003. ACM.

[23] C. C. Venters, R. J. Hartley, M. D. Cooper, and W. T. Hewitt. Query by visual example: Assessing the usability of content-based image retrieval system user interfaces. In *PCM '01: Proceedings of the Second IEEE Pacific Rim Conference on Multimedia*, pages 514–521, London, UK, 2001. Springer-Verlag.

[24] M. Worring, C. G. M. Snoek, O. de Rooij, G. P. Nguyen, and A. W. M. Smeulders. The MediaMill semantic video search engine. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages –, Honolulu, Hawaii, USA, April 2007. *Invited paper*.

[25] E. Zavesky and S.-F. Chang. Cuzero: embracing the frontier of interactive visual search for informed users. In *MIR '08: Proceeding of the 1st ACM international conference on Multimedia information retrieval*, pages 237–244, New York, NY, USA, 2008. ACM.