JPSearch: an Answer to the Lack of Standardization in Mobile Image Retrieval

Frederik Temmermans\textsuperscript{a,b}, Mario Döller\textsuperscript{c}, Iris Vanhamel\textsuperscript{d}, Bart Jansen\textsuperscript{a,b}, Adrian Munteanu\textsuperscript{a,b}, Peter Schelkens\textsuperscript{a,b}

\textsuperscript{a}Vrije Universiteit Brussel (VUB), Dept. of Electronics and Informatics (ETRO), Pleinlaan 2, B-1050 Brussels, BELGIUM
\textsuperscript{b}Interdisciplinary Institute for Broadband Technology (IBBT), Dept. of Future Media and Imaging (FMI), Gaston Crommenlaan 8 (box 102), B-9050 Ghent, BELGIUM
\textsuperscript{c}University of Passau, Innstrasse 34, 94034 Passau, GERMANY
\textsuperscript{d}Universum Digitalis, Pakhuisstraat 32, 1000 Brussels, BELGIUM

Abstract

The amount of imagery that is available via various content providers grows at a staggering rate. Contemporary handheld devices allow accessing these content providers anytime, anyplace. Moreover, using the build-in camera, one can easily contribute to these online image repositories. The present lack of consistency in how repositories are accessed for retrieval or syncing complicates inter-operability between devices and systems. JPSearch, a recent standardization initiative within the Joint Photographic Experts Group (JPEG) committee, resolves this predicament by defining the interfaces and protocols for data exchange between them, while restricting as little as possible how those devices, systems or components perform their task.

Keywords: interoperability, standardization, JPSearch, mobile search

1. Introduction

An exploding amount of pictures is available via various online image sharing applications, which we will further refer to as content providers. Contemporary
handheld devices allow accessing these content providers anytime, anyplace. Many mobile applications can perform this task. For example, Apple’s App Store contains over 100 applications, only to access Flickr. When looking for a particular image it makes sense searching through more than one of those content providers. However, most content providers employ dedicated Application Programming Interfaces (API’s) to access their content. As a consequence, third party search applications can only support to access a predefined set of content providers. A second major problem relates to image, and image repository annotation. Contemporary online and offline photo management applications do not use a standardized way to manage and exchange those annotations with other systems [1, 2]. The considerable investment of annotating images often is lost, when images are transferred from one system to another. This leads to yet another problem: the lack of a standardized way to synchronize multiple online and/or offline repositories. All those problems have been identified by the JPEG committee and led to the JPSearch initiative. JPSearch defines the components of a framework for interoperability of still image search and retrieval. More specific, it defines interfaces and protocols for data exchange between devices and systems, while restricting as little as possible how those devices, systems or components perform their task [3].

This article is structured as follows. Section 2 emphasizes the need for standardization in this domain, both from a business as from a user’s perspective. Next, this section gives an overview of common applications and concepts. Finally, it describes the current problems in image retrieval and repository management and synchronization. Section 3 contextualizes the JPSearch standard within the JPEG committee and gives a detailed overview of the standard. Section 4 gives an overview of how JPSearch differs from MPEG-7 and how both standards complement each other. Section 5 demonstrates how interfacing with JPSearch compliant repositories works. Section 6 gives an overview of strategies and design patterns, necessary to implement the fundamentals of a JPSearch based mobile application. Section 7 explains how to deal with more advanced queries to support techniques like query by picture, using MPEG-7 descriptors.
2. Need for standardization in mobile image retrieval

2.1. The mobile application market

Lately, mobile devices became a major source for image generation. For example, counting the number of people who uploaded at least one image with a particular camera to Flickr, shows that Apple’s iPhone exceeds by far popular DSLR cameras like Nikon’s D90\(^1\). This trend, in conjunction with the existence of powerful mobile operating systems, and the success of application distribution models, such as the Google Android Market, opens an exciting and innovative market for mobile applications relating to image sharing and retrieval.

Handheld devices differ significantly from desktop computers in hardware and usage. Including support for the various sensors inherent to mobile devices, e.g. cameras, GPS, accelerometers, digital compasses, instigates innovative applications. However, device limitations in terms of memory, processing power, and power consumption necessitate particular consideration. Smaller and touch enabled screens prompt for a dedicated approach in user interface design. While desktop applications typically run for a relatively long period, we turn mobile applications on and off consistently. Human impatience dictates slick startup times and efficient processing. Applications relying on mobile data connectivity definitely include the optimization of bandwidth usage in their design.

New business opportunities arise from firm smartphone sales and the success of mobile application distribution mechanisms, such as Apple’s App Store. However, a growing market that rapidly changes and evolves, challenges developers considerably. Moreover, a vast competition leads to saturated markets in which unit prices are relatively low. An application should stand out from the crowd in this competitive market and appeal to the masses. As such, the continuous integration of the latest technologies maintains a competitive product. However, a viable business case also takes future concepts and technologies into account and tries to retain backwards compatibility as long as possible. The

\(^1\)http://www.flickr.com/cameras/
latter ensures reaching sufficient potential users.

Many image sharing or retrieval applications are part of a client-server architecture in which they represent a client node as shown in Figure 1. Typically, these applications are build upon different tiers or layers. The highest tier is the presentation tier, which contains the Graphical User Interface (GUI). It connects via the subsequent logic tier to a data representation tier. Many applications rely on a fourth tier that connects to a server to send and retrieve data, i.e. the service-connecting tier. Changes to the upper two layers, e.g. improving the user interface or introducing new features, are directly perceived by the user. This is less the case for the lower two layers. On the other hand, since the application’s performance heavily depends on the lower two layers, investing an appropriate amount of time on these two is a necessity. Being able to reuse those components in multiple applications, or the ability to rely on open source packages, improves cost-effectiveness. Therefore, standardization at this level can be a huge help, as we will discuss in more detail later on.

2.2. Mobile image retrieval and sharing

Mobile image retrieval and sharing applications contain image retrieval, image sharing functionality or both.

Sharing allows the immediate distribution of personal photos or other image like content. Popular distribution channels are online repositories (Flickr, Pi-
casa, etc) and social websites (Facebook, Netlog, etc). In this context, sharing also includes synchronization between the mobile application and a desktop application, such as Adobe’s Lightroom or Apple’s iPhoto. An important aspect of sharing is that the same image can be stored using different file formats, resolutions and compression levels. This is subject to characteristics of the client application and/or repository. Metadata and tags usually describe the image’s content. Some metadata is automatically generated, e.g. geo tagging registers where the photo is taken. The image creator can add information such as copyright, title and much more. Repositories may also modify, add or even remove content descriptors. Finally, communities often tag shared data. All this extra information facilitates retrieval.

Image retrieval in mobile applications serves a wide variety of purposes. Commonplace text searches result in a set of relevant images and adopt one or more keywords to match content descriptors. The support for logic operators and/or regular expression instigates sophisticated queries. In such a case, the objective is retrieving images. The concept of query by picture uses an image for generating query input sometimes in conjunction with text search. In addition to retrieving similar images, information related the object of interest can be returned, e.g. Google Goggles. Nowadays, mobile devices have other sensors that can be used in the retrieval process. For example using location and orientation has advantages for the user, because it reduces the amount of irrelevant results. Moreover, it optimizes the retrieval process because it reduces the initial search space for the query-by-picture process. Another less trivial input generator is the touch screen of a mobile phone. The definition of Regions Of Interest (ROI) can refine the image input. Query by sketch is another concept that uses the touch screen.

2.3. Why Standardization

Currently, interoperability concerns between applications and distribution channels confront both users and developers choosing carefully which systems to adopt/support. From the user’s point of view, changing repository is not as
straightforward as it could be. In addition, consulting more than one distribution channel concurrently is still not a simple task. The business side also demands some careful consideration, because supporting multiple distribution channels currently remains a daunting task. Common interoperability issues are:

- **Interfacing**: Application Programming Interfaces (APIs) form the basis of accessing most contemporary online image repositories. They adopt standardized formats such as XML, JSON, etc. However, because no standardized schema exists, developers face implementing a different service connector for each repository they wish to support in their application. Hence accessing multiple repositories from a single application remains complicated and inefficient. This issue clearly highlights the importance of adopting multi-tier architecture in which only the base layers are affected.

- **Advanced queries**: The support for contemporary and forthcoming sophisticated queries remains an open issue. Advanced searching techniques for images are a hot research topic. As these methods mature quickly, they will become available for wide usage. Hence, applications that include such features turn out to be highly desirable.

- **Metadata schemas**: The storage of metadata can be contained within the image file, in a separate file or both. Many different metadata schemas exist. Popular examples include EXIF [4], Dublin Core [5, 6] and XMP [7]. Moreover, dedicated schemas comprise domain specific fields. In the case of medical imaging metadata, DICOM [8] specifies its own metadata schema. Remark that simultaneous use of different metadata schemas is possible.

- **Exchanging annotations**: Most repositories allow their users to annotate images and collections of images. Annotations offer a great way to navigate and search through repositories. Annotations can sometimes be made either by the owner, the community or be auto-generated. Annotating large collections of images takes a considerable amount of time. Therefore, it is
important that the annotations are kept when the images are transferred to other repositories. Currently, this is not always the case, especially collection annotations are mostly not retained.

- **Importing, exporting and synchronizing repositories** Many people manage multiple repositories. These repositories include online and offline repositories and images or collections of images managed on mobile devices. There is no standardized way of how collections can be imported or exported from one system to another, or how multiple repositories can be synchronized with each other.

  Proper standardization overcomes the above predicaments. Clearly, the problem is not the manner in which individual components perform their tasks in the framework. The difficulty resides in the way different components communicate with each other. JPSearch focuses on standardizing these communication components.

3. **JPSearch**

JPSearch (ISO/IEC 24800) is a recent standardization initiative within the Joint Photographic Experts Group (JPEG) committee, formally known as ISO/IEC JTC1 SC29 WG 1. The JPEG committee is a joint working group of the International Standardization Organization (ISO) and the International Electrotechnical Commission (IEC) [9, 10]. JPSearch aims at providing a standard for interoperability of still image search and retrieval systems. It provides an abstract search framework architecture that contains a set of interfaces and protocols. It seeks for decoupling the general tightly coupled image search systems [11].

query format whilst Part 4 [15] adopts the JPEG and JPEG 2000 image file formats for embedding metadata information. Part 5 [16] defines an interface to synchronize multiple repositories, or to move a repository from one system to another. Finally, Part 6 [17] introduces reference software. Most parts are standards in progress and are expected to become international standards in the near future. The following sections cover the individual parts in more detail.

3.1. Part 1: System architecture

![Figure 2: Components of the JPSearch standard](image)

Part 1 gives a general overview of the complete system framework and its components. It outlines the functionality and purpose of all components within the JPSearch framework. Figure 2 shows the general architecture and how the other parts relate to each other. Due to its informal character, Part 1 is published as a technical report [12]. This report also defines several use cases which should be supported by the final framework. These use cases help illustrating the scope of JPSearch, therefore, some of them are mentioned hereafter:
• Searching images in stock photo collections for usage in magazines: A user wishes to buy a selection of images in order to illustrate a publication to be sold to consumers.

• Mobile tourist information: A tourist is in an unfamiliar place, sees an interesting landmark and wants to know what it is. He takes a picture of the landmark on his mobile phone and sends it to a tourist information server which calls him back and gives him the information.

• Surveillance search from desktop to mobile device with alerts: The user sets up a visual surveillance query on a desktop computer, saves the query for real time monitoring with results saved periodically for retrieval from a mobile device.

• Finding illegal or unauthorized use of images: The user holds his original content. He wants to find unauthorized variations of his original content using search engines.

• Matching images between collections for synchronization: The user ends up with a large collection of images stored on multiple computers, laptops, external drives, portable USB key drives, portable photo players, photo sharing websites, photo printing websites, and mobile phones. He needs his collection to be synchronized across all platforms.

• Image search in the medical domain: Many pathologies have visual symptoms which are essential for doctors doing diagnosis. There are cases where the visual symptoms are not familiar and the doctor would like to consult visual reference material (an atlas, or previously diagnosed reference cases). The user (a doctor in a hospital clinic) searches then for the best matches to the symptoms and retrieves case histories (including other images, metadata, text, etc.) to aid his/her diagnosis.

• Servants image searchers: The user accesses content by means of a standard web browser. The website forwards the user’s requests to the application using a standard communication protocol. For each provided
query, the application creates a personalized intelligent search agent that will effectively take care of the search on behalf of the user.

3.2. Part 2: Registration, identification and management of schema and ontology

Part 2 of JPSearch consists of three main areas. The JPSearch Core Schema serves as root schema for any image description and should be understood by any retrieval system. The translation mechanism provides means for mapping descriptive information between various XML based metadata models, and finally the management unit deals with registered metadata models and its translation rules.

3.2.1. JPSearch Core Schema

The JPSearch Core Schema is modeled as an XML schema and provides a light-weight and extensible mechanism for annotating image content. The current version supports 20 main elements including spatial and region of interest based annotation. The individual properties have been chosen according to their coverage by popular metadata formats. A similar approach of selecting the core properties was also chosen by the W3C Media Annotations group\(^2\) for their *Ontology for Media Resource*\(^3\) media matching framework. Furthermore, the JP-Core schema features an extensibility mechanism for integrating parts of other formats by using the `ExternalDescription` element.

Code 1 demonstrates a simple image annotation. It contains administrative information, like `Creators` or `CreationDate`, and basic descriptive annotations about the content, such as `Description` and `Keyword`. It also provides technical information about the size (`Width` and `Height`). Finally, it shows how specific salient and interesting regions within an image are denoted, by using the `RegionOfInterest` element. This element supports the specification of events, persons or places.

\(^2\)http://www.w3.org/2008/WebVideo/Annotations/
\(^3\)http://dev.w3.org/2008/video/mediaset/mediaset-1.0/mediaset-1.0.html
3.2.2. Translation Mechanism

The Translation Rules Declaration Language (TRDL) provides means for translating descriptive information within JPSearch queries to and from the JPSearch Core Schema into equivalent information of a respective target schema.

It is important to note that TRDL focuses on mappings at the structural and syntactic level for transforming the descriptive content of queries. A practical example is the mapping of the compact Creators field of the Dublin Core schema to the MPEG-7’s creator’s family and given name elements, as illustrated in Figure 3.

In general, the transformation formalism is based on a set of transformation rules. The syntax and semantic of every rule follows one out of three different transformation types. The first one is the one-to-one transformation type. This
Figure 3: Difference between the specification of a creator in MPEG-7 and Dublin Core

<table>
<thead>
<tr>
<th>Core Schema</th>
<th>MPEG-7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>CreationInformation → Creation → Title</td>
</tr>
<tr>
<td>Keyword</td>
<td>CreationInformation → Creation → Abstract</td>
</tr>
<tr>
<td>CreationDate</td>
<td>CreationInformation → Creation → CreationCoordinates → Date</td>
</tr>
<tr>
<td>Creators</td>
<td>CreationInformation → Creation → Creator (Role=creator)</td>
</tr>
<tr>
<td>Publisher</td>
<td>CreationInformation → Creation → Creator (Role=publisher)</td>
</tr>
</tbody>
</table>

Table 1: Excerpt of core schema elements with their equivalent MPEG-7 elements

type is used if the according attributes or elements only differ in their tag name. The other two types are many-to-one and one-to-many. Many-to-one will be applied, when the metadata information must be transferred from a set of source attributes and/or elements into a single target field. One-to-many works vice versa to many-to-one. Each transformation type consists of one or more FromField elements and one or more ToField elements. The FromField elements specify the core schema or the fields to which the original condition refers. The ToField elements specify the mappings needed to generate one or more new conditions over the external format which are equivalent to the original ones.

In order to tackle semantic content transformations, like splits or merges of certain fields, the TRDL provides the ability to declare regular expressions over the content of elements or attributes. A more detailed description can be found in [18].

The process of mapping descriptive information between two different metadata models starts by identifying matching elements and attributes. For instance, Table 1 presents an assignment of some JPSearch Core elements and
respective elements of the MPEG-7 standards. Based on the identified matching elements, translation routines are defined for every element and attribute. Code 2 gives an example of the one-to-many transformation rule for the *Creators* element.

**Code 2** Example one-to-many transformation rule for *Creators*.

```
<TransformationRules>
  <TransformationRule type="OneToManyFieldTransformationType">
    <FromField type="FilteredSourceFieldType">
      <XPathExpression>//*[a-zA-Z]*</XPathExpression>
      <FilterWithRegExpr>([a-zA-Z]*/[a-zA-Z]+)</FilterWithRegExpr>
      <VariableBinding>$a</VariableBinding>
    </FromField>
    <ToField type="TargetFieldType">
      <XPathExpression>../../CreationInformation/Creation/Creator/Agent/Name/GivenName/$a</XPathExpression>
    </ToField>
    <ToField type="TargetFieldType">
      <XPathExpression>../../CreationInformation/Creation/Creator/Agent/Name/FamilyName/$b</XPathExpression>
    </ToField>
  </TransformationRule>
</TransformationRules>
```

3.2.3. Registration Authority

The management of metadata models and their interplay is, due to its diversity and heterogeneous character, an important aspect of the standard. Therefore, the JPSearch project will establish a set of registration authorities which will keep a list of metadata schemas, together with their related translation rules. The main tasks of the JPSearch registration authorities are as follows:

- Establish simple registration procedures according to directives of the hosting standardization body.
• Provide a mechanism for an easy identification of metadata schemas and their translation rules.

• Establish procedures for the management and verification of registered information.

Currently, the process of finding registration authorities has been started [19]. The final list of registration authorities will be listed by ISO4.

3.3. Part 3: Query format

Part 3 defines the JPSearch Query Format (JPQF). It is a common standardized message protocol, that facilitates searching across multiple repositories by using the same search semantics. This message protocol defines a set of input parameters, to describe the search criteria, and a set of output parameters, to describe the aggregated result sets for user presentation or machine consumption. The query format supports advanced search techniques, including query by example and relevance feedback. The JPQF relies on the features of the MPEG Query Format (MPQF) restricted to the image domain [20].

The normative parts of the JPQF define three main components. The Input Query Format provides means for describing query requests from a client to an image repository. The Output Query Format specifies a message container for retrieval responses. Finally, the Query Management Tools arrange all organizational aspects of sending a query to a repository. This includes functionalities such as service discovery, service aggregation and service capability description.

An Input Query can be composed of three different parts: a declaration, an output description and a query condition. The declaration contains references to resources, which can be used within the output description or query condition. Such a reference can for example point to an image file or a metadata description file. The output description allows to define the structure as well as the content of the expected result set. Finally, the query condition denotes

4http://www.iso.org/iso/maintenance_agencies
the search criteria. Therefore, it provides a set of different query types (e.g., QueryByMedia) and expressions (e.g., GreaterThan), which can be combined by fuzzy boolean operators. Within a query, XPath expressions can be used to address elements and attributes in a metadata schema.

Section 7 contains some examples of JPQF queries.

3.4. Part 4: File format for metadata embedded in image data

Part 4 adopts the JPEG and JPEG 2000 image file formats for embedding metadata information [9]. The benefit of integrating and combining metadata and raw image data is the mobility of the metadata, and its persistent association with the image itself. Data compliant with Part 4 can be circulated among any JPSearch compliant system without loss of information. As a consequence, users do not have to repeat time-consuming image annotation tasks. It also benefits developers by minimizing the effort of implementing metadata updating functions, e.g. when the image data is modified by an image editing tool. An example implementation of the file format has been integrated in the image annotation tool Caliph [21]. An initial version of this implementation is available on the IRIS website5.

3.5. Part 5: Data interchange format between image repositories

The scope of Part 5 is to standardize a format for exchanging image collections and associated metadata between JPSearch compliant repositories. The metadata may be at the level of an image or at the level of an image collection. The JPSearch data interchange format facilitates the interoperable synchronization, import and export of repositories. Adoption of this part should result in the following advantages for users:

- Easy exchange or synchronization of data between repositories on different devices and platforms.
- Leveraging the generally high cost of creating metadata.

5http://www.dimis.fim.uni-passau.de/iris/index.php?view=jpsearch
• Painless transfer of data to a newer and better system.
• Consolidating metadata generated on different systems.
• Consolidating selected data to a centralized repository.
• Archiving data in a format that will survive current products.

4. JPSearch vs MPEG-7

This section gives an overview of how JPSearch differs from MPEG-7 and how both standards complement each other. For this purpose, first the MPEG-7 standard is briefly introduced in Subsection 4.1 which is followed by an analysis of its interplay with JPSearch in Subsection 4.2.

4.1. MPEG-7

MPEG-7, formally known as ISO/IEC 15938 - Multimedia Content Description Interface, is the ISO/IEC international multimedia description standard developed by MPEG (Moving Picture Experts Group). MPEG-7 concentrates on describing multimedia content in a semantically rich manner. While earlier MPEG versions were focusing on making multimedia content available, MPEG 7 allows the localization, filtering, managing and processing of desired media files by tagging them with information regarding their content and origin [22, 23]. From a technical point of view, MPEG-7 standardizes Descriptors (D) and Description Schemes (DS). The former defines the syntax and semantics of each feature representation while the latter specifies relationships between components (both descriptors and description schemes). The Description Definition Language (DDL) allows defining and modifying both DS and D. The DDL is based on an XML schema, extended by new data types, for example, for feature vector representation. In addition, Classification Schemes (CS) support the extensibility of MPEG-7. This includes parental rating, and content classification into a number of pre-defined categories. Currently, MPEG-7 is organized in 12 different parts. In the following a short overview of some parts is provided:
- **Part 1 - Systems:** This Part includes tools for preparing MPEG-7 descriptions to allow an efficient transport and storage (BiM - binary format for MPEG-7). Further, it defines mechanisms for supporting synchronization between content and their descriptions. Additionally, the MPEG-7 Systems component provides tools for managing and protecting intellectual property.

- **Part 2 - Description Definition Language (DDL):** The DDL is one of the core parts, used for instantiating MPEG-7 descriptions. It provides a descriptive foundation for creating user defined description schemes and descriptions.

- **Part 3/4 - Visual/Audio:** These description tools consist of basic description schemes and descriptors that mainly cover visual (like color, texture) or audio (like melodic, spoken content) information.

- **Part 5 - Multimedia Description Schemes (MDS):** MDS standardizes on the one hand generic descriptors and descriptor schemes that are common to all media (e.g. vector) and on the other hand multimedia entities that represent more complex structures (e.g. for content management).

- **Part 11 - Profiling:** The profiling part provides the possibility to restrict descriptions of the MPEG-7 schema.

- **Part 12 - MPEG Query Format:** This is the latest achievement and has been standardized in 2008. The query format specifies the access to multimedia retrieval systems.

To demonstrate the use of MPEG-7 for describing multimedia content, Code 3 shows an annotation for the soccer game image shown in Figure 4. The image is decomposed into two sub-images. A *StillRegion* descriptor specifies each sub-image. The annotation defines for each sub-image an ID, e.g. *midfieldTactics*, a *TextAnnotation* that contains text description of the image, and some low level feature instances, e.g. *VisualDescription.*
4.2. Differences and Interplay

Although at first sight, the two standards target on different use cases, they have a lot in common. MPEG-7 supports the annotation of any kind of multimedia data via a rich set of descriptors and description schemes that both cope with low and high level features. Additionally, MPEG-7 handles administrative or content management information. In contrast, JPSearch targets the specification of image retrieval systems and their interaction. This entails that JPSearch does not focus on annotating content but on annotation exchange. In order to get a clear picture, the individual parts of JPSearch are discussed in the viewpoint of MPEG-7.

− **Part 2:** As introduced in section 3 this part of JPSearch has three main areas: the JPCore schema, translation rules declaration language and their management facilities. Most important for comparison with MPEG-7 are the first two ones. Similar to MPEG-7, the JPCore schema defines a set of elements for annotating image content. Besides some global information (like title, keywords or description), the main focus of JPCore is directed to the annotation of regions of interest and their content (like occurring persons or events). The core schema has been developed by investigating available image metadata formats and selecting those elements and corresponding semantic concepts that show a broad overlap.

MPEG-7 provides a very complete description format that covers any media type. Consequently, JPCore is, as for instance Dublin Core, a subset
of the MPEG-7 schema. Clearly, all information in a JPCore related instance document is expressible as an MPEG-7 instance document. However, unlike MPEG-7, JPSearch specified the JPCore schema to be lightweight, omitting the known complexity of MPEG-7. For instance, the facilities that describe the semantic information of salient regions are reduced to a few important elements. Although JPCore has a lightweight design, its extensibility mechanism allows integrating features of other metadata formats. For example, the well-formalized visual low-level features of MPEG-7 [24] can be adapted into JPCore descriptions. In this way, image retrieval systems can support content-based annotation and retrieval, based on visual features,
using MPEG-7.

The second area in Part 2 considers interoperability among metadata formats. The translation language for metadata formats supports distributed image retrieval scenarios over multiple heterogeneous repositories. The availability of manually created translation guidelines between the MPEG-7 format and the JPCore makes possible an automated mapping between MPEG-7 and JPCore metadata during retrieval or during annotating metadata within the file format. The JPSearch reference software [25] contains definitions of translation instances between MPEG-7 and JPCore. It demonstrates the interaction of the translation rules with Part 3 and Part 4 of the standard.

- **Part 3:** The JPSearch Query Format has probably the largest correlation with the MPEG-7 standard, simply, because the JPSearch query format is a subset of the MPEG Query Format [20], i.e. the MPEG-7 subset relevant to the domain of still image search.

- **Part 4:** This part specifies a new file format, based on the JPEG/JPEG 2000 coding schemes [9]. This file format allows storing and transmitting the combination of image data with its associated metadata as a single file. The format introduces a *Metadata block* which can contain multiple *Elementary Metadata blocks*. The first *Elementary Metadata block* is reserved for holding the JPCore Schema, subsequent *Elementary Metadata blocks* are usable for other XML based metadata formats.

Two options exist for integrating MPEG-7 metadata. The first option stores the descriptive information in a single MPEG-7 instance document, which is then stored in a separate Elementary Metadata block after the obligatory JPCore Elementary Metadata block. This first block can be left empty. However, it advisable filling it with mapped metadata by using appropriate translation rules between MPEG-7 and JPCore. Reason being that all JPSearch compliant systems guarantee the support of JPCore. However, duplicating information can lead to inconsistencies. Hence, the second and
recommended option incorporates parts of MPEG-7 via the extensibility feature of the JPCore schema. For example, descriptors for visual low-level features that are not covered by the JPCore schema can be included in this way.

In order to support an efficient coding behavior of the XML content, it is foreseen to use the binarisation mechanism of MPEG-7 systems [26].

- **Part 5:** Part 5 defines a data interchange format for supporting synchronization between image repositories. The synchronization covers item level as well as collection level data. Initially, this part of the standard focuses on the use of JPEG/JPEG 2000 coded image data and their enhancements, specified in Part 4. Therefore, all relationships between JPSearch and MPEG-7 presented before are valid as well. In addition to a container for the image data, a *ItemMetaBoxType* type has been defined. This type is meant to keep item level information. Similar as for Part 4, it is recommended expressing the metadata information by the JPCore schema, although other metadata formats, including MPEG-7, are supported as well.

5. **Interfacing with JPSearch Compliant Systems**

JPSearch compliant clients access repositories directly or circuitously. If the repository is a JPSearch compliant system, direct communication with the client application is possible. Alternatively, a JPSearch compliant system can function as an intermediate and aggregate data from one or more third party repositories.

Both methodologies are illustrated further. First, we examine the main components of a JPSearch compliant service and browse through their interplay.

- **Core Schema:** It serves as metadata basis supporting interoperability during search among multiple image retrieval systems. Clients use the core schema for formulating search requests in JPEG Query Format (see Section 3).
- **Transformation Rules:** These rules map the metadata of a query of an incoming search request to compatible search requests of the native systems.

- **Metadata Registration authority:** This body administers and distributes metadata ontologies and their transformation rules. It is hosted by JPEG.

- **JPQF Query Reformulation:** Once the metadata is compliant with the target, the query is subject to an optional reformulation. This may be due to the nature of the new metadata schema or for any other reason, e.g. user preferences.

- **JPQF Aggregation Service:** This service joins the respective results of the individual queries sent to multiple repositories.

Figure 5: Query process to a JPSearch compliant system.

Figure 5 illustrates the workflow of a simple system where a client application
queries a native JPSearch compliant system. First, the JPSearch compliant system receives the JPQF input query from the client application (1). Then, the system converts the metadata, in the core schema, to a native schema, using the transformation rules (2). Optionally, the JPQF query is reformulated (3). Subsequently, the interpreter translates the JPQF query to the native query language (4). The translated query is executed on the database. Finally, the metadata of the results is transformed back to the core schema (5).

Figure 6: Query process to a JPSearch compliant system that aggregates data from multiple third party repositories.

Figure 6 illustrates a workflow where a JPSearch compliant client accesses an intermediate JPSearch compliant system, which in its turn communicates with multiple third party repositories. Again, the JPSearch compliant client sends a JPQF input query to the intermediate system that will retrieve results from N
third party systems (1). The metadata based on the core schema is transformed to N native schemas (2). If necessary a query reformulation is applied N times (3). For each third party repository, an interpreter exists, which can translate the JPQF query to the native query language of the corresponding third party repository (4). The metadata of the N individual result sets are converted to the core schema (5) and the N results sets are aggregated to a single result set (6). The aggregated result set is forwarded to the user.

6. Mobile development strategies and design patterns

6.1. Handling XML

As mentioned in Section 2, implementing parts of the JPSearch standard mainly affects the lower layers of a multitier application, namely, the tiers that are responsible for data representation and connecting to the backend. These layers represent the foundation of the application. Consequently, the application’s performance heavily depends on the implementation of these layers. In the case of a JPSearch based mobile application, these layers include one or more of the following processes: query creation, generation, parsing, validation, representation and exchange. Figure 7 represents their interplay. JPSearch’s
components are commonly based upon XML. Thus, handling XML efficiently is paramount. Two common ways of parsing XML exist: Document Object Model (DOM)\(^6\) and Simple API for XML (SAX)\(^7\).

DOM parses the complete XML file and returns an object model representation, which is accessible via various accessor methods. SAX fires events whilst parsing the file, e.g. an event occurs for every opening or closing tag. For both strategies, many implementations exist in many languages and for different platforms. While DOM parsing has been popular because of its simplicity, it has some major disadvantages with respect to memory management and validation amongst others. Large XML files necessitate relatively large amounts of memory, because the complete XML file is loaded into an object model. Moreover, the DOM parser’s nature does not encourage memory management by developers.

SAX parsing typically requires more effort on the part of the developer. However, the increased control makes it a more suitable choice, especially in the case of mobile applications that handle large and/or many XML files. Apart from implications regarding memory usage, SAX allows validation during parsing. An invalid element can immediately interrupt the parsing process and throw a warning or exception. Some implementations, like libxml2 \(^8\), are able to parse the data while it is still in transfer. This allows interrupting both parsing and downloading as soon as an invalid element is encountered. Data parsed with a SAX parser usually is represented in an object model too. In contrast with DOM, this object model is tailored to the XML file’s structure. Moreover, this custom object can also be used for generation, e.g. to build a query to send to a retrieval system.

\(^6\)http://www.w3.org/DOM/
\(^7\)http://www.saxproject.org/
\(^8\)http://www.xmlsoft.org
6.2. Query by picture

An implementation of the query by picture concept comprises preprocessing, feature extraction and matching. In an initial step, preprocessing reduces undesirables, like noise, whilst enhancing the desirables in the raw image. The type of preprocessing highly depends on the application since sought-after features vary with purpose. For example, when querying a database of paintings, the preprocessing step includes a foreground extraction that separates the painting from the background. The subsequent step characterizes the processed image or object of interest by computing features and expressing them in a feature vector. The final step matches this feature vector with feature vectors stored in the database. The matching process uses some kind of distance measure, e.g. Euclidian distance. The result is either an exact match or a set of similar images. The type of results depends on the request or application. Remark that for each image present in the consulted database, the same type of feature vector is computed beforehand and can hence be stored in a structure suited for efficient matching.

![Diagram of Query by Picture Implementation]

Figure 8: Two strategies for implementing Query by Picture. Saving client processing (top) or saving bandwidth (bottom).

When implementing the query by picture concept for a mobile application, two basic strategies, illustrated in Figure 8 are common practice. One option sends solely the feature vector to the server. Hence, it entails that the mobile client executes preprocessing and feature extraction. Alternatively, sending the
picture to the server ensures that the server performs the entire processing chain. The former optimizes bandwidth while the latter minimizes processing and memory usage. The optimal strategy clearly depends on the complexity of the specific use case. The use of lightweight features favors adopting the first strategy, while for computational intensive features the second option may result in better efficiency. A third possibility is an on the fly methodology in which the circumstances dictate which mechanism should be adopted. For example, the presence of a faster Wi-Fi connection prompts sending the picture to the server, while a 3G or EDGE connection results in calculating some lightweight features and sending these to the server. Note that in some cases sending the image itself also improves compatibility, because the server is not restricted in its choice of features.

7. Adopting JPSearch to an existing system

7.1. MoidEx

The MoidEx application, recently introduced in [27], is an example of a query-by-picture application using the bandwidth saving strategy introduced in the previous section. This application is a location based mobile tourism (mTourism) system, that supports tourists in receiving information about an object, such as a building in a foreign country. Figure 9 shows the overall architecture of the system.
On the client side, the application extracts several low level MPEG-7 features, including the Scalable Color Descriptor, Edge Histogram Descriptor, Dominant Color Descriptor [24]. Although the Color Layout Descriptor and the Edge Histogram Descriptor both contain information on the spatial distribution of colors and edges, this information is still very coarse and might not be sufficient to reliably recognize the specific structure of e.g. a church. In crowded places, occlusion by other objects, like cars or buses, and different weather conditions might cause additional difficulties. Therefore, in order to support the identification of buildings in an image, the MoideEx system makes also use of the Consistent Line Cluster (CLC) algorithm [28].

The CLC algorithm is based on the fact that the shape of buildings and other man-made objects typically contain many straight line segments, which come from the boundaries of windows, doors, or the building itself, as shown in Figure 10. Another important observation is that if two line segments belong to different objects, the local colors around them are usually different. Moreover, line segments from different objects most likely belong to different spatial groups.

The MoideEx system describes the extracted information (low level features and the CLC) as an MPEG-7 instance document. The basic idea is to store the actual contents of the clusters, instead of the final CLC histogram. This approach has two advantages: On the one hand, it reduces computational costs on the mobile devices. On the other hand, this also facilitates the use of other feature vectors besides the CLC histogram.

Given a consistent line cluster $c_l_i$ with lines:

$$l_1 = ((x_{11}, y_{11})(x_{12}, y_{12})), \ldots, l_n = ((x_{n1}, y_{n1})(x_{n2}, y_{n2}))$$
the proposed mapping of this cluster to the MPEG-7 RegionLocatorType
descriptor is shown in Listening 1.

Listing 1: Mapping of line clusters to polygons

```xml
<SpatialLocator xsi:type="RegionLocatorType">
  <Polygon>
    <Coords mpeg7:dim="4 * n">
      x1 1 1 1
      y1 1 1 1
      xn 1 2 2
      yn 2 2 2
    </Coords>
  </Polygon>
</SpatialLocator>
```

Finally, the extracted information is sent to the mTourism library, which
is build on top of the MPEG-7 MMDB [29]. In this scenario, the geolocation
database contains images showing landmarks and a description of their exact
geographic position. Then, the CBIR service detects the best matching images
within the geolocation database on the basis of a nearest neighbor search among
the visual low level features and its location.

7.2. Adopting JPSearch

Although the MoidEx system uses MPEG-7 descriptors, it does not employ
a standardized way for client-server communication. As a consequence, third
party mobile applications, that can extract those descriptors, can still not com-
municate with the MoidEx backend. On the other hand, the MoidEx client is
also incapable of consulting third party repositories that support those specific
MPEG-7 features. This predicament is common in most contemporary CBIR
systems, and this is exactly what the JPSearch initiative tries to remedy.

To be compatible with JPSearch compliant clients, first of all, the system
needs to support the JPSearch Query Format. Remark that a server can support
multiple services. This is useful for intermediate repositories as presented in
Section 5. All those services have a unique identifier (ServiceID). The MoidEx
example supports one service, with service identifier CBIR_Mobile. The first
step a JPSearch compliant client does, is checking the supported services and
their capabilities. The management part of JPQF defines how this can be done.
If the client knows the service ID, it can query for its capabilities with the query given in Code 4.

**Code 4** Example input management query

```xml
<?xml version="1.0" encoding="UTF-8"?>
<jpqf:JPEGQuery>
  <jpqf:Management>
    <jpqf:Input>
      <jpqf:ServiceID>CBIR, Mobile</jpqf:ServiceID>
    </jpqf:Input>
  </jpqf:Management>
</jpqf:JPEGQuery>
```

When the client does not specify any service ID, the server returns a list of all services and their respective capabilities. Alternatively, the client can specify some desired capabilities. In this case, the server will only return those services which support the requested capabilities.

**Code 5** Example of an output management query.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<jpqf:JPEGQuery>
  <jpqf:Management>
    <jpqf:Output>
      <jpqf:AvailableCapability serviceID="CBIR, Mobile">
        <jpqf:SupportedMetadata>urn:mpeg:mpeg7:schema:2004</jpqf:SupportedMetadata>
        <jpqf:SupportedQueryTypes href="urn:...:ServiceCapabilityCS:2008:100.3.6.1"/>
        <jpqf:SupportedQueryTypes href="urn:...:ServiceCapabilityCS:2008:100.3.6.2"/>
        <jpqf:SupportedQueryTypes href="urn:...:ServiceCapabilityCS:2008:100.3.2.1"/>
        <jpqf:SupportedQueryTypes href="urn:...:ServiceCapabilityCS:2008:100.3.2.2"/>
        <jpqf:SupportedQueryTypes href="urn:...:ServiceCapabilityCS:2008:100.3.2.3"/>
      </jpqf:AvailableCapability>
    </jpqf:Output>
  </jpqf:Management>
</jpqf:JPEGQuery>
```

Code 5 shows the output query the MoidEx server would return. It uses the MPEG-7 metadata model (SupportedMetadata) and is able to evaluate the QueryByMedia (100.3.6.1) and QueryByDescription (100.3.6.2) query types. Additionally, it can use the logic operators AND, OR, NOT, respectively 100.3.2.1, 100.3.2.2, 100.3.2.3 and any combination of them.

Once the client is aware of the server’s capabilities, it can commence sending input queries. For MoidEx, the client application extracts low level features
based on the MPEG-7 standard. Code 6 shows an example of a query that uses the `ScalableColor` and `EdgeHistogram` features, which are combined by a logical `AND` and assigned with preferences indicating that color is more important than edges. Note that, due to the fact that the JPSearch Query Format is a subset of the MPEG Query Format, a JPQF query can contain both namespaces, as shown in the example.

**Code 6 Example JPSearch Query Format request**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<jpqf:JPEGQuery jpqfID="Query_1">
  <jpqf:InputQuery>
    <jpqf:QueryCondition>
      <jpqf:Condition xsi:type="mpqf:AND">
        <mpqf:Condition xsi:type="mpqf:QueryByDescription">
          matchType="similar" preferenceValue="80">
            <mpqf:DescriptionResource resourceID="ID_1">
              <mpqf:AnyDescription xmlns:mp7="urn:mpeg:mpeg7:schema:2004">
                <mpqf:Mpeg7>
                  <mp7:DescriptionUnit xsi:type="mp7:ScalableColorType">
                    numOfCoeff="16" numOfBitplanesDiscarded="0">
                      <mp7:Coeff>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16</mp7:Coeff>
                    </mp7:DescriptionUnit>
                  </mp7:Mpeg7>
                </mpqf:Mpeg7>
              </mpqf:AnyDescription>
            </mpqf:DescriptionResource>
          </mpqf:Condition>
        </mpqf:Condition>
        <mpqf:Condition xsi:type="mpqf:QueryByDescription">
          matchType="similar" preferenceValue="20">
            <mpqf:DescriptionResource resourceID="ID_2">
              <mpqf:AnyDescription xmlns:mp7="urn:mpeg:mpeg7:schema:2004">
                <mpqf:Mpeg7>
                  <mp7:DescriptionUnit xsi:type="mp7:EdgeHistogramType">
                    <mp7:BinCounts>1 2 3 ...</mp7:BinCounts>
                  </mp7:DescriptionUnit>
                </mp7:Mpeg7>
              </mpqf:AnyDescription>
            </mpqf:DescriptionResource>
          </mpqf:Condition>
        </mpqf:Condition>
      </jpqf:Condition>
    </jpqf:QueryCondition>
  </jpqf:InputQuery>
</jpqf:JPEGQuery>
```

Finally, after evaluation of the request, the server codes the results as an output description of JPQF, containing individual `ResultItems`.
8. Conclusions

Image search and repository management on online image repositories is hindered by different aspects. Most content providers define their own API’s for users or third party applications to query the image database; no standardized way is used to exchange metadata; and syncing between multiple repositories works different for each application. The growing market of smartphones forms yet another major platform used for image search, image generation and image sharing or management. Many mobile applications are available to perform those tasks. All of them are restricted to work only with a certain set of content providers. JPSearch identified those problems and defines interfaces and protocols which can lead to interoperability. However, JPSearch does not restrict the different system components in how they perform their task. Presently, most parts of JPSearch are standards in progress, but they are expected to become international standards in the near future. At that moment, it is up to the content providers to recognize the need and the advantages for them to adopt those standardized interfaces.

Acknowledgment

This work is supported in part by the Fund for Scientific Research - Flanders (FWO) by the postdoctoral fellowships of A.Munteanu and P.Schelkens, the Interdisciplinary Institute for Broadband Technology and by Innoviris.

References


http://www.itscj.ippsj.or.jp/sc29/29w42901.htm


