Artificial Intelligence and Query Execution Methods in the VizIR Framework

Doris Divotkey, Horst Eidenberger, Roman Divotkey

Vienna University of Technology
Institute of Software Technology and Interactive Systems
A-1040 Vienna, Favoritenstrasse 9-11/1882, AUSTRIA
{doris.divotkey, horst.eidenberger, roman.divotkey}@ims.tuwien.ac.at

Abstract

The article introduces the architecture of the querying components of the visual information retrieval framework VizIR. A major design goal was to assure adaptability and extensibility in manifold ways. VizIR components can be arbitrarily combined to build extensive applications. The framework provides various visual content descriptors, similarity measures and query models. Moreover, the platform can be extended by visual features or similarity measures as well as entirely novel query paradigms. VizIR introduces an approach for Video Browsing based on MPEG-7 visual features and Self-Organizing Maps for clustering. Furthermore, a recently proposed approach for integrating browsing and retrieval techniques is presented. Following this approach, the user is enabled to interact with the querying system in several ways which improves retrieval quality and performance considerably.

Introduction

Querying large databases of multimedia data is a major domain within content-based image and video retrieval research. A lot of visual information retrieval platforms confine to supporting only one querying model (e.g. *k*-nearest neighbor). Therefore research on investigating new approaches in querying can hardly be done with those systems. It is desirable to have an open platform for arbitrary query models that employ arbitrary feature descriptors and similarity measures.

VizIR is intended to be a workbench for research: highly flexible, adaptable and extensible in many different ways. It is easy to configure the platform for different requirements. One possible way to do so is to build applications by taking ready-made components of the VizIR system. In addition, the framework is open to be extended by new components. In case of querying these could be additional feature descriptors, novel similarity measures or entirely different query models.

The article is organized as follows: The next section gives an overview on the VizIR project in general, followed by a description of the state of the art in visual information retrieval. We describe approaches followed in VizIR where intelligent learning algorithms are applied. Furthermore, the querying architecture of VizIR and some implementation details are presented.

Background: The VizIR framework

VizIR is an open and freely available framework for content-based image and video retrieval. Developed in Java, it provides a huge collection of classes for major visual information retrieval tasks like storing and managing media objects and the corresponding descriptor data. It contains a strong and extensible querying and retrieval component with algorithms for automatic feature extraction, similarity measurement and graphical user interfaces for browsing the media databases, query formulation (by example or sketch) and query refinement. The coupling between user interfaces and querying components is loosely defined. The same user interfaces may be used for different query engines as well as user interfaces can be exchanged or adopted easily. The communication among the components is based on XML messages that can either be applied directly, or the mapped Java classes (embedded in the system) can be used. The framework provides implementations of various descriptors for image and video data (including the MPEG-7 visual descriptors).

Among others, the VizIR framework consists of the following main parts (system architecture is shown in Figure 1):

- The service kernel is responsible for the communication between the components and for the management of queries, query execution, evaluation and so forth. The communication between the kernel and the user interface for query formulation is based on XML messages, e.g. the multimedia retrieval language (MRML) can be used [9]. In addition, a simple and easy to adopt querying language has been defined to be open for future extensions (this is presented later in more detail). The kernel initiates the conversion of the incoming XML queries to appropriate Java-classes that may be handled by the query engine.
- The strong and easily extensible querying and retrieval component defines a simple interface fulfilling the requirements of any query model. Moreover, the querying component provides an abstract base class implementing the above mentioned interface. The base class offers several methods that are of general interest. Further information on the querying architecture and implementation details are presented in the next subsections.
- Furthermore, the framework provides user interfaces for management of media objects and feature descriptors, for query formulation, presentation of the result set and support of query refinement. In addition, tools for visualizing descriptor data are available.
- The *PersistenceSystem* is a database management layer for efficient storage of media objects and feature descriptors. [3]

VizIR is released under the GNU General Public License, source code and documentation can be received from the project webpage under [11]. Further information on the VizIR project can be found in [2].

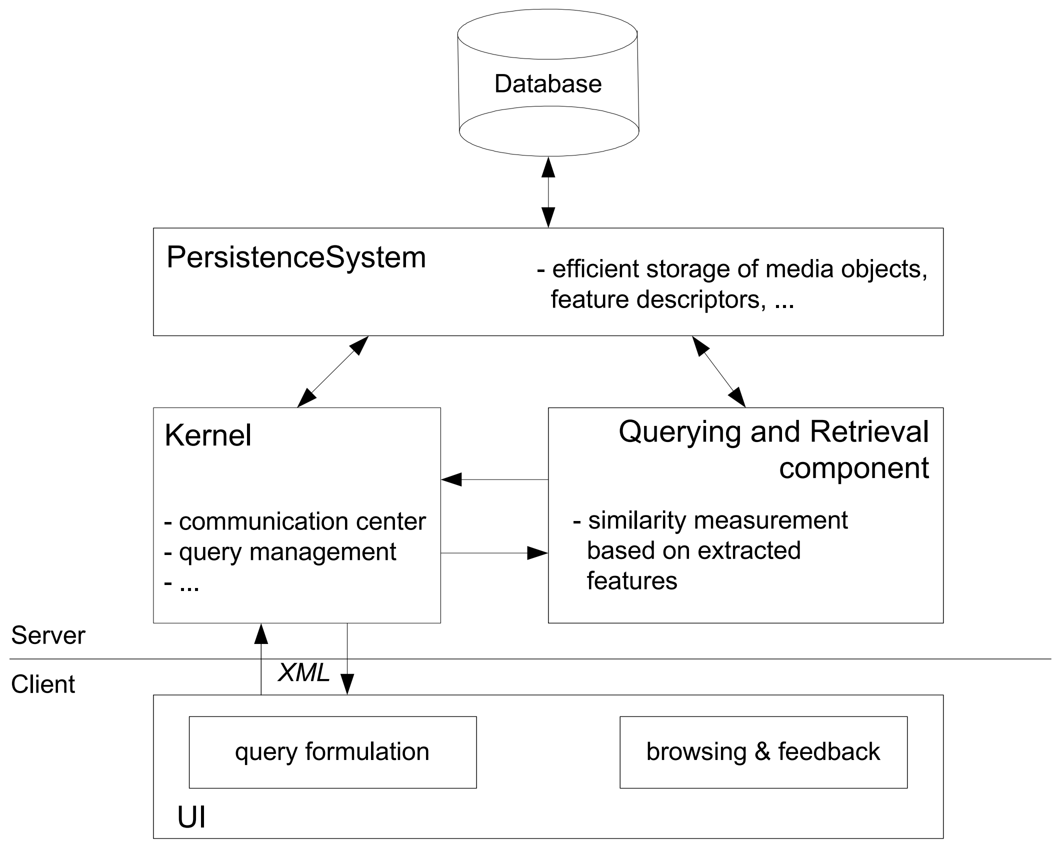
Database
PersistenceSystem
- efficient storage of media objects, feature descriptors, ...
Kernel
- communication center
- query management
- ...
Querying and Retrieval component
- similarity measurement based on extracted features
Server
Client
XML
query formulation
browsing & feedback
UI

Figure 1. System architecture of the VizIR Retrieval Framework

Querying models and retrieval techniques

State of the art in visual information retrieval

Content-based visual information retrieval bases on extracting visual features and determining similarity of images or videos according to these characteristics. Common non-probabilistic techniques treat media objects as vectors in a feature vector space. The distance of two points within the given vector space measures the similarity of the corresponding media. Mostly, it is assumed that the feature space conforms to a metric space in which the metric axioms can be implied.

Despite considerable research in visual information retrieval in recent years, the retrieval results can only be considered as partially satisfactory (in highly domain-specific applications). Among other reasons this is due to the difficulty of expressing the semantic content (high-level concepts) of images or videos with low-level features (semantic gap) [1]. The following points have significant effect on the retrieval result:

- Selection of appropriate features that describe the visual content sufficiently

- Selection of distance measures for determining the similarity between features and images respectively
- Decision on a method to use for calculating a single similarity measure from the high-dimensional distance vector (e.g. in case of the *k*-nearest neighbor approach this is the sum of the weighted distance values which leads to the problem to find appropriate weights for the individual features)

Determining and testing settings for these degrees of freedom is a challenging task of retrieval applications. Although a vast amount of different approaches have been proposed the semantic gap could so far not be closed. Besides, the above mentioned assumption, that the feature space is a metric space may have negative effect on the retrieval outcome as well. Several cases have shown that the human perception of similarity does not follow the metric axioms. Moreover, subjectivity is of great importance (but as well a limitation) of computer-centric retrieval approaches. User-centered techniques that regard the user as an important actor within the retrieval process perform significantly better. Based on user feedback, techniques like Bayesian networks, Self-Organizing Maps or Kernel-based Learning are employed to improve the retrieval performance. [7,9]

Artificial Intelligence techniques used in VizIR

This subsection gives an outline of Artificial Intelligence techniques used in VizIR. These techniques comprise learning algorithms for intelligent user interfaces and for clustering and analysis purposes.

Clustering based on MPEG-7 visual descriptors

Among others, the MPEG-7 standard defines a set of descriptors for visual media that are state of the art in visual information retrieval [8]. These descriptors are used to extract characteristics of the media content and to determine the similarity of media objects. Clustering of media objects according to their visual features is performed in various retrieval and browsing applications. In this respect, several clustering techniques have been used. We decided on Self-Organizing Maps as they produce a rather “natural” clustering compared to human perception. The SOM [7] is a two-layer fully connected neural network that uses feed-forward learning. The input layer is interpreted as a one-dimensional data vector; the output layer as a two-dimensional map (clusters). Each cluster of the output map is described by a vector of connection weights pointing to its center (codebook vector). In training and application, input data vectors are mapped to the codebook vector with minimum Euclidean distance (best matching unit, BMU). One major innovation of SOMs over other clustering methods is the introduction of neighborhood kernels. These two-dimensional functions define the fraction, to which the BMU is adapted but also, to which extent neighboring codebook vectors are adapted. Thus, SOM learning means learning of cluster neighborhoods.

In a study we analyzed the efficiency of MPEG-7 descriptors. Among others, a cluster analysis was performed on the extracted features. Thereby, Self-Organizing Maps were applied to cluster the media collections based on the derived MPEG-7 visual descriptors. In SOMs the high-dimensional feature vectors are mapped to a two-dimensional grid representing the clusters. During learning stage, the media objects are reallocated based on their distances to the individual codebook vectors. The usage of appropriate neighborhood kernels results in clusters that match nicely with human perception of similarity. A typical neighborhood kernel is a Gaussian-like function. The SOM clustering provides a topological view of the data as it shows the relationships of elements and clusters. Hence, cluster analysis brings up valuable information. It indicates redundancies of elements and detects holes between element clusters. [6]

Visual Data Mining

In the context of the VizIR project a novel approach for interactive content-based retrieval of visual information has been proposed (see also [4]). The main aspect there is to integrate browsing and retrieval to improve efficiency of the retrieval process and the retrieval results in particular. The basic idea is to include the user in the retrieval cycle in a more direct and flexible way than traditional approaches do (like e.g. query refinement by relevance feedback). The user interface presents the media objects in a three dimensional space and provides the user with several possibilities of interaction like the definition of hyper-clusters and neighborhoods (spherical areas within distance spaces). Furthermore, the user can adjust the position of media objects in the feature space simply by drag and drop interaction. As the whole process is highly user-driven, it is up to the user to signify the result as sufficient.

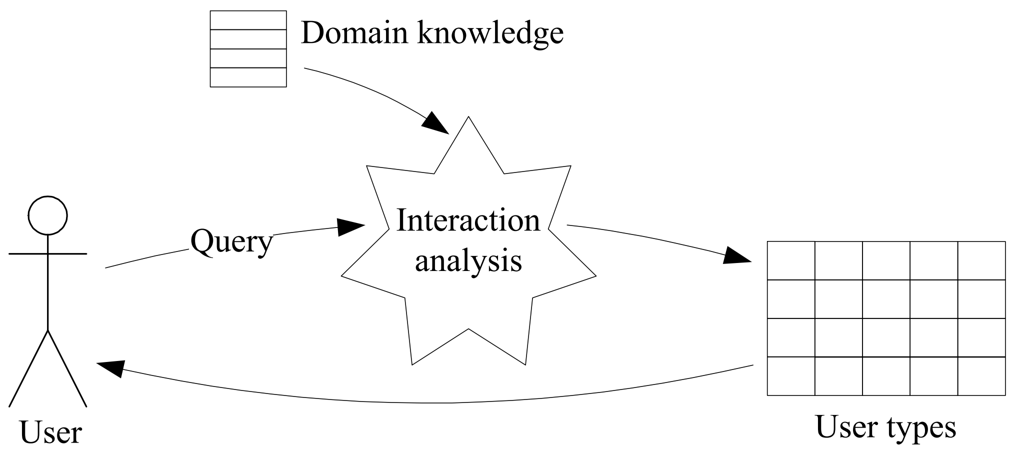

Figure 2. User learning and application cycle

The analysis of user's querying behavior and the provision of tailor-made visualizations helps improving the retrieval quality. Figure 2 shows the learning and application cycle for user interaction. The system learns user types by deriving statistical indicators based on the users' querying behavior. According to this information the application provides suggestions to the user and updates its knowledge. User learning helps to facilitate the selection of features and distance measures and the definition of clusters.

Video Browsing

A novel way to facilitate the access of video content has been achieved by an interactive video browsing approach that indexes video streams both by time and by content [5]. The indexing is done hierarchically whereby the levels are of different granularity: lower levels present more details than higher ones. Two separate index trees for time and content are generated. The user can switch between the two different views at any time. Switching is performed by a matching procedure between the two index trees. Video segments and shots are represented by key-frames. For the time-index tree, key-frames are chosen according to the temporal order of the video frames. The content-index tree is built from the shots of the video stream.

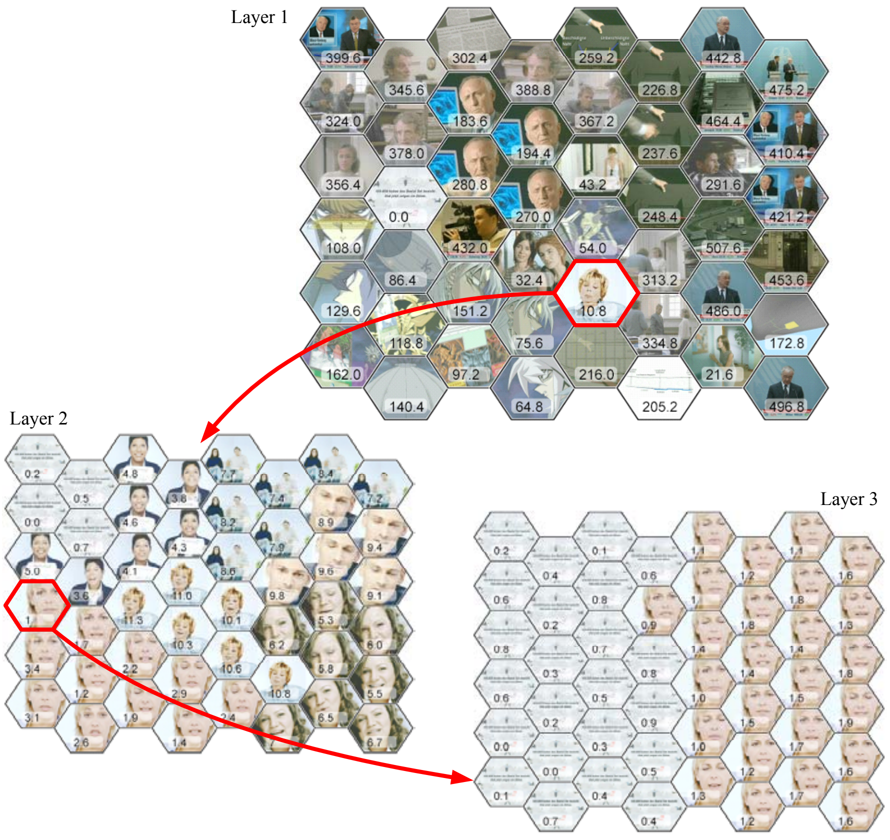

Figure 3. Example screenshot of time index view. The figure shows maps on three layers. Layer 1 is the top layer computed from the test videos used in the evaluation

In detail, MPEG-7 descriptions of the key-frames are extracted and clustered using Self-Organizing Maps. The result is a two-dimensional map of similarity-based clustered key-frames for each level of both index trees (as visualized in Figure 3). Each cluster is represented by the cluster center vector. The input vectors are assigned to the closest codebook vectors (the best matching units). In the training stage the system adapts the codebook vectors until the sum of distances to all input vectors (normalized by the number of vectors) is minimal.

SOM clustering of content-based features provides a spatial organization of data based on the most significant visual stimuli of media. These are: color, texture and shape. Since these features are in particular important for human perception, SOM clustering provides valuable results for fast and effective video browsing.

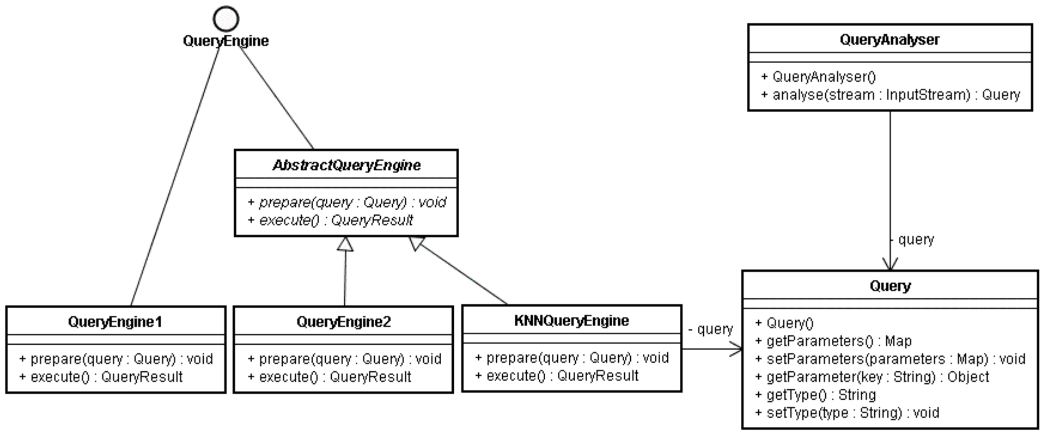
QueryEngine
AbstractQueryEngine
+ prepare(query : Query) : void
+ execute() : QueryResult
QueryEngine1
+ prepare(query : Query) : void
+ execute() : QueryResult
QueryEngine2
+ prepare(query : Query) : void
+ execute() : QueryResult
KNNQueryEngine
+ prepare(query : Query) : void
+ execute() : QueryResult
- query
QueryAnalyser
+ QueryAnalyser()
+ analyse(stream : InputStream) : Query
- query
Query
+ Query()
+ getParameters() : Map
+ setParameters(parameters : Map) : void
+ getParameter(key : String) : Object
+ getType() : String
+ setType(type : String) : void

Figure 4. Class diagram of the VizIR query engine (simplified)

Querying architecture

The major design issue of the VizIR querying component was to establish a flexible architecture for provision of a possibility to adapt and extend the system according to special requirements like simple exchange of feature descriptors for evaluation and benchmarking purpose as well as adding a new query model for research. Besides, the system is open to support relevance feedback, query refinement, parallel and distributed querying.

The querying component of the VizIR framework is based on the common content-based visual information retrieval paradigm. Visual information of media is extracted and the similarity of media is calculated due to distance measures in the feature space. Arbitrary models can be applied for combining several features. For instance, VizIR offers an implementation of the *k*-nearest neighbor approach. According to an example image or video it retrieves the *k* best matching media by minimizing the feature distance functions in consideration of the given feature weights.

Implementation

Query engine

Figure 4 shows the class structure of the querying framework. The interface *QueryEngine* takes a key position, it defines the two methods query engines have to support to be integrated into the framework. The abstract base class provides the developer with methods of common interest for a wide range of query models. Generally, there are two ways to create a new querying engine for the VizIR framework. The first is to implement the interface *QueryEngine* and the second is to derive the abstract base class and override the methods *prepare()* and *execute()*. The latter has the advantage that one can rely on methods defined by the base class like convenient methods for easy access of descriptor data, for sorting the result set, etc.

In *prepare()* the query engine initializes and configures itself for the proximate query execution. The actual query execution takes place within the *execute()*-method. The

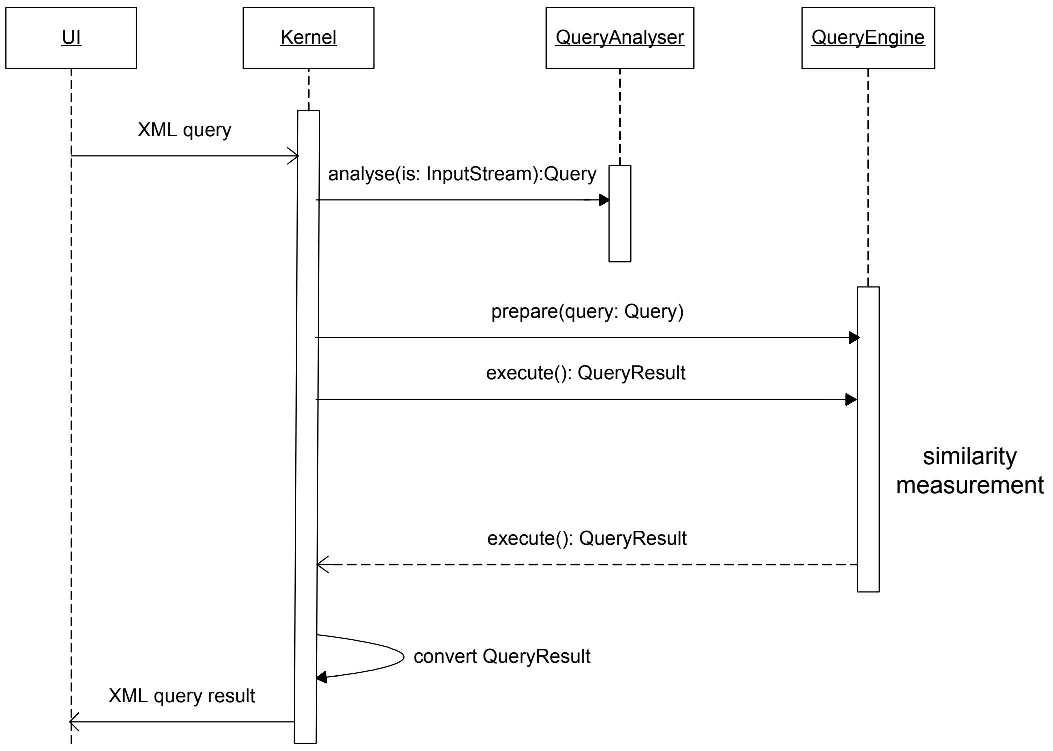

Figure 5. Query execution

media are retrieved from the database, with the corresponding descriptor data the distances to the example media are calculated. The query engine receives an instance of the class *Query* (passed to the method *prepare()* as parameter). The *Query* class has a *type* flag. Hence, the interoperability can be checked easily. *Query* contains the parameters necessary for query execution.

Query engines may have arbitrary query parameters, which results in the need of an open structure. To provide the required flexibility, the XML messages for query formulation follow a generic approach (see next section). The class *QueryAnalyser* converts the XML query messages to the Java class *Query*, that holds the parameters in a map. The corresponding instance of *QueryEngine* is capable of extracting und applying the specified parameters. The retrieval result is kept in a class *QueryResult*, containing the individual query result entries given by the unique identifier of the media and the according similarity measure for this object. The class that initiated the query execution (e.g. the service kernel) may cause the conversion of *QueryResult* back to XML. Then, the XML messages can be transferred to the user interfaces for further processing. Figure 5 shows the process of query execution within the VizIR architecture.

Query formulation in XML

The VizIR framework supports the multimedia retrieval language (MRML) for query formulation [10]. Since MRML satisfies only few query models, the VizIR querying framework provides a simple, but highly flexible XML specification that allows formulation of queries tailor-made for any query engine.

The Document Type Definition looks as follows:

```
<!ELEMENT query (param+, paramlist*)>
<!ATTLIST query
      type CDATA #REQUIRED
>
<!ELEMENT param (name, value)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT value (#PCDATA)>
<!ELEMENT paramlist (name, item+)>
<!ELEMENT item (param+)>
```

A query consists of single parameters (e.g. example media or result size) and parameter lists (like the list of feature descriptors including the appendant weights). Furthermore, a query has the attribute *type* for verifying type compatibility. A *param* is a tuple of name and value. The *paramlist* may consist of several items containing an arbitrary amount of parameters.

Below, an example XML-message for a *k*-nearest neighbor query is given:

```
<query type="KNN">
  <param>
    <name>example</name>
    <value>imgid</value>
  </param>
  <param>
    <name>resultSize</name>
    <value>30</value>
  </param>
  <paramlist>
    <name>descriptors</name>
    <item>
      <param>
        <name>descriptorId</name>
        <value>1</value>
      </param>
      <param>
        <name>weight</name>
        <value>0.5</value>
      </param>
    </item>
    <item>
      <param>
        <name>descriptorId</name>
        <value>2</value>
      </param>
      <param>
        <name>weight</name>
        <value>0.5</value>
      </param>
    </item>
  </paramlist>
</query>
```

The query stated above contains the following parameters: *example* refers to the image used for query by example. It is given by the identifier *imgid*. The *resultSize* is 30 at most. The parameter list includes the features used for similarity measurement. In this case two descriptors are applied, each of them is weighed with 0.5. In the Java representation, the parameters are kept in a *HashMap*. The keys are given by the names of the parameters. With this simple way to represent query parameters extending the VizIR framework by new query models can be done easily and with little effort.

Conclusion and future work

Visual information retrieval research is in need of a workbench that provides sets of context specific components and methods. Furthermore, it is desirable to easily adopt and extend the framework for facilitating the integration and evaluation of new approaches. VizIR is one answer to these demands. The flexibility of a main part of the framework – the querying component – is presented. Future work will include the further development and extension of querying models with substantial interest in user-centric approaches. The focus will lie in developing and improving strategies to learn about the user's needs and preferences.

Acknowledgements

The VizIR project is funded by the Austrian Scientific Research Fund (FWF) under grant no. P16111-N05.

References

[1] A. Del Bimbo Visual Information Retrieval, Morgan Kaufmann, San Francisco CA, 1999

[2] H. Eidenberger, C.Breiteneder (2003) VizIR – A Framework for Visual Information Retrieval. Journal of Visual Languages and Computing 14, 443-469

[3] H. Eidenberger, R. Divotkey (2004) A Data Management Layer for Visual Information Retrieval, Proceedings ACM Multimedia Data Mining Workshop, Seattle WA, 48-51

[4] H. Eidenberger (2004) Visual Data Mining, SPIE Information Technology and Communication Symposium (Internet Multimedia Management Systems), Philadelphia PN, 121-132

[5] H. Eidenberger (2004) A Video Browsing Application Based on Visual MPEG-7 Descriptors and Self-Organizing Maps, International Journal of Fuzzy Systems, Vol. 6, No. 3

[6] H. Eidenberger (2004) Statistical analysis of MPEG-7 image descriptions, ACM Multimedia Systems journal, Springer, Vol. 10, No. 2, pp. 84-97

[7] T. Kohonen (1990) The Self-Organizing Maps, Proceedings of the IEEE, 78/9, 1464-1480

[8] B.S. Manjunath, P. Salembier, T. Sikora (2002) Introduction to MPEG-7, Wiley, San Francisco CA

[9] Y. Rui, T.S. Huang (2001) Relevance Feedback Techniques in Image Retrieval, in M.S. Lew (ed.) Principles of Visual Information Retrieval, Springer, Heidelberg, 219-258

[10] University of Geneva, MRML Project Webpage (last visited 2005-03-12), http://www.mrml.net

[11] Vienna University of Technology, VizIR Project Webpage (last visited 2005-03-12), http://vizir.ims.tuwien.ac.at