

VizIR—a framework for visual information retrieval

Horst Eidenberger*, Christian Breiteneder

Interactive Media Systems Group, Institute of Software Technology and Interactive Systems, Vienna University of Technology, Favoritenstrasse 9-11-188/2, A-1040 Vienna, Austria

Received 31 May 2002; received in revised form 4 March 2003; accepted 28 April 2003

Abstract

In this paper the visual information retrieval project VizIR is presented. The goal of the project is the implementation of an open visual information retrieval (VIR) prototype as basis for further research on major problems of VIR. The motivation behind VizIR is the implementation of an open platform for supporting and facilitating research, teaching, the exchange of research results and research cooperation in the field in general. The availability of this platform could make cooperation and such research (especially for smaller institutions) easier. The intention of this paper is to inform interested researchers about the VizIR project and its design and to invite people to participate in the design and implementation process. We describe the goals of the VizIR project, the intended design of the querying framework, the user interface design and major implementation issues. The querying framework consists of classes for feature extraction, similarity measurement, media handling and database access. User interface design includes a description of visual components and their class structure, the communication between panels and the communication between visual components and query engines. The latter is based on the multimedia retrieval markup language (MRML, Website: <http://www.mrml.net> (last visited: 2003-03-20)). To be compatible with our querying paradigm, we extend MRML with additional elements. Implementation issues include a sketch on advantages and drawbacks of existing cross-platform media processing frameworks: Java Media Framework, OpenML and DirectX/DirectShow and details on the Java components used for user interface implementation, 3D graphics with Java and Java XML parsing.

© 2003 Elsevier Ltd. All rights reserved.

Keywords: Visual information retrieval; Content-based image retrieval; Content-based video retrieval; Media processing

*Corresponding author. Tel.: +43-158801-18853; fax: +43-158801-18898.

E-mail addresses: eidenberger@ims.tuwien.ac.at (H. Eidenberger), breiteneder@ims.tuwien.ac.at (C. Breiteneder).

1. Introduction

The global integration of information systems with the ability of easy creation and digitization of visual content have risen the problem of how these vast amounts of data in collections or databases are managed. One of the crucial success factors of all approaches to solve this problem is apparently the implementation of effective but still easy to handle retrieval methods. Visual information retrieval (VIR) is still a rather new approach to overcome these problems by deriving features (or: descriptors; like color histograms, etc.) from the visual content and comparing visual objects by measuring the distance of features with distance functions. VIR is usually divided in two directions: Content-based image retrieval (CBIR) and content-based video retrieval (CBVR). The major advantages are fully automated indexing and the description of visual content by visual features. On the other hand, the fundamental drawbacks of VIR are the *semantic gap* between high level concepts presented to a user and the low level features that are actually used for querying [1] and the *subjectivity of human perception*. The latter means that different persons or the same person in different situations may judge visual content differently. This occurs in various situations: different persons may judge features (color, texture, etc.) differently, or if they judge them in the same way they still may perceive them differently [2].

Partly due to these principle drawbacks four major problems of VIR approaches can be identified:

- Low result quality.
- Complicated user interfaces.
- Unsatisfactory querying performance.
- Lack of assessment methods.

Retrieval results of low quality are—next to semantic gap and subjectivity of human perception—often the consequence of working only with general features for all types of visual content and asking the user to choose the features, he would like to use. Complicated user interfaces overtax the casual user if they demand for a precise opinion on similarity, the selection of features, and especially, the provision of feature weights. Many users would not even try a classic VIR interface, if they had the opportunity to use it. Simpler, but still effective user interfaces are needed to improve the acceptance of VIR systems.

Unsatisfactory querying performance—especially for large media collections—is a result of using distance functions in VIR systems to calculate the dissimilarity between visual objects. This process is often very slow and unbearable reply times may occur for large databases. Query acceleration methods include (1) indexing techniques (e.g. R*-trees), (2) complexity reduction techniques (e.g. coarse feature vector representation or suitable transformations) and (3) media object occlusion techniques (e.g. using the triangle inequality in [3]).

Finally, despite reasonable efforts in the last 3 years, very few standardized methods exist for assessing new querying paradigms. One exception is the Brodatz

sample collection, which represents some kind of de-facto standard for the evaluation of texture querying. Promising approaches to overcome this situation are the Benchathlon project [4] that tries to collect and compare the performance of CBIR benchmarks and the annual TREC video retrieval competition [5] that defines evaluation procedures for CBVR.

In this paper we present the visual information retrieval project VizIR. The goal of the project is an open VIR framework as a basis for further research in order to overcome the problems pointed out above. The basic structure of VizIR was first laid down in [6,7]. VizIR was initiated in summer 2001 as a consequence of our experiences gained in earlier VIR projects and is supported by the Austrian research fund since December 2002. The motivation behind VizIR is: an open VIR platform would make research (especially for smaller institutions) easier and more efficient (because of standardized evaluation sets and measures, etc.). The intention of the paper is to let interested researchers know about the VizIR project and its design and to invite them to participate in the design and implementation process.

The goal of VizIR is not the development of a monolithic system but of a system-independent class framework of querying and user interface components (interaction panels, event model, etc.) based on the Java programming language. An important issue of VIR is the communication of user interfaces and query engines. This communication should be standardized in order to combine arbitrary user interfaces and querying systems and be based on modern communication paradigms (XML, etc.).

The paper is organized as follows: the following section points out relevant related work, Section 3 is dedicated to the VizIR project goals and Section 4 to the querying and user interface framework design. Section 5 discusses major implementation issues and finally, Section 6 gives an overview over past, current and next activities in the VizIR implementation process. The paper is supplemented by an appendix with an extension of the MRML [8].

2. Related work

In this section we discuss the architectural properties and shortcomings of earlier CBIR and CBVR prototypes and the user interface approaches that were used.

2.1. Existing VIR prototypes

Past research efforts have lead to several general-purpose prototypes like QBIC [9], Virage [10], VisualSEEK [11], Photobook [12], MARS [13], El Niño [1,14] and GIFT [15] for CBIR as well as OVID [16] and VIQS for CBVR and some application-specific prototypes like image retrieval systems for trademarks [17] or CueVideo for news videos analysis (e.g. [18]). Most of these prototypes share a number of serious drawbacks. The first is that all of them implement only a small number of features and do not offer the developer an API for extensions. An

exception is IBM's QBIC system for image querying, which has (in version 3) a well-documented API for feature programming.

Another problem is that none of these prototypes has an architecture that supports the MPEG-7 standard (see [19]). To our knowledge, at present no MPEG-7-compliant prototype for VIR exists or is under development. Part 6 of MPEG-7 contains a reference implementation of its visual descriptors and a simple querying application, which was developed for testing and simulation [19]. Unfortunately, this reference implementation does not contain a framework, a documentation of the VIR part, a modern user interface (though a simple web-interface for experts is available by now), a suitable database, optimized descriptor extraction functions and performance-optimized algorithms. That is why it cannot be used as a VIR prototype, although it is still a good starting point for developing one.

One prototype that should be mentioned here is the GNU image finding tool (GIFT). GIFT is an extendible CBIR system (developed at the University of Geneva) available under GNU public license [15]. Unfortunately, GIFT supports only image querying and because it is based on C++ and the Unix operating system it can not be extended to video retrieval easily. Currently, no standardized video processing environment with a C/C++-API is available for Unix operating systems (see Section 5.1). Still, GIFT introduced several valuable concepts to CBIR (including MRML, see Section 2.2).

Apart from the mentioned focal points of research and the implemented prototypes the following *key issues* of VIR systems have not yet been investigated to a sufficient extent:

- Similarity measurement in multi-feature environments.
- Media sets for assessment.
- Integration of computer vision methods.

With similarity measurement we mean the transformation of a distance space (the result of distance measurement for multiple features and distance functions) to a result set. The common way of similarity measurement in VIR systems is measuring distances with an L^1 - or L^2 -metric (e.g. city block distance and Euclidean distance), merging a single object's distance values for multiple features by the weighted sum and presenting the user the objects with the lowest distance sum as the most similar ones. We have shown in our earlier work that this approach is not the most effective one [20]. More sophisticated methods for similarity definition would result in higher quality results (e.g. [21]).

Additionally, as pointed out above, not enough effort has been undertaken so far to put together standardized rated image and video sets for the various groups of features. This has led to vague, often worthless statements on the quality of VIR prototypes.

Finally, surprisingly few ideas and methods have been taken over from computer vision and other areas up to now. Neural networks have been used for feature clustering (e.g. self-organizing maps [43]), face detection and thresholding methods for segmentation but hardly any shaping techniques for 3D object reconstruction or sophisticated neural networks for scene analysis have been yet applied.

2.2. VIR user interfaces

This section overviews user interfaces of well-known VIR systems: first CBIR systems and then CBVR systems. The focus in CBIR will be on classic systems (including QBIC and Virage) and two promising more recent approaches (El Niño and ImageGrouper). The section ends with a short description of an approach to standardize the communication of VIR user interfaces and query engines.

In the past, the design of user interfaces of VIR systems was quite simple—in comparison to most other visual systems. Most systems (QBIC, Virage [10], Photobook, VisualSEEk) use a single 2D panel of images for query definition and result set display. Querying is done by selecting one or more query examples, one (e.g. QBIC), a few (e.g. MARS) or all features (Virage) and—in the latter two cases—weights for the importance of these features. Iterative Refinement by Relevance Feedback [44,45] can usually be performed by defining the importance of result set elements textually and iterating the query. This paradigm has several drawbacks: earlier result sets are thrown away, selecting features and weights overtaxes the casual user and after all, the static structure of such an interface is not very user-friendly and from today's point of view may be judged old-fashioned.

Therefore several research groups have been working on new user-centric interface approaches in the last years. Two of the most interesting are El Niño and ImageGrouper [22]. To our knowledge, El Niño is the first approach to define a query implicitly by the distance relations of objects in a 3D panel. This query definition process can be done intuitively and easily by drag-and-drop. The most interesting innovations in ImageGrouper are the usage of two panels for the active and the last query and a history over all refinement steps in a querying session. The central idea of ImageGrouper is the definition of queries by three groups: positive examples, negative examples and neutral examples. ImageGrouper's major drawback is that it has no standard interface to query engines and is bound to an engine with classic distance measurement and linear weighted merging.

Like El Niño, VizIR will contain 3D user interfaces for query formulation. Using 3D information visualization techniques instead of 2D methods has several advantages. Generally, each 3D view is just a 2D projection [23]. 3D views take advantage of human spatial memory and allow displaying more information without incurring additional cognitive load because of pre-attentive processing of perspective views. In general, they lead to better retrieval results in user studies in terms of reaction time, number of incorrect retrievals and failed trials [24]. Additionally, they allow the rendering of more information items because of scaling possibilities and a better global view. Finally, there is experimental evidence that 3D displays enhance subjects' spatial performances [23]. The major open problem of 3D systems in this context is the development of suitable 3D user interaction techniques [24,25].

Classic CBVR systems are OVID [16] and VQIS. One of the most interesting aspects concerning the user interfaces of CBVR systems is the handling of temporal media (video and animations) in a static user interface. In general, there are three

principle solutions to present video information: (1) integration of the full video with player controls into the environment (CPU power and network bandwidth consuming), (2) creation and usage of animated icons (CPU power consuming) and (3) creation of still images that represent the video content. The third solution is the most widely applied one (in VIR). The simplest form of the third type is an image matrix of all keyframes in a video clip. Another approach is the Micon, a 3D cube showing the first frame of a video clip as well as the first line and the last column of all consecutive frames (see element A and B in Fig. 5 for examples). Another type is the Hierarchical Video Browser, a tree-structured view of a video clip. In [26] a general overview of different presentation styles for video is given.

The interoperability of VIR user interfaces and querying systems is an issue that is gaining more and more attention. Interoperability should be achieved by standardized interfaces. The most promising effort in this direction is the MRML (developed at the University of Geneva [8]). MRML is an XML-based standard. It is implemented in GIFT, the user interface Charmer and the basis of the Benchathlon project (see [8] for details). We try to incorporate MRML into the user interface components of VizIR.

3. VizIR project goals

This section gives an overview of the objectives of the VizIR project. VizIR aims at the following major goals:

- Implementation of an open VIR class framework.
- Integration of MPEG-7 visual.
- Implementation of a framework of user interface components for VIR.
- Support for distributed querying.

The overall goal is the implementation of a modern, *open class framework* for content-based retrieval of visual information as basis for further research on successful methods for automated information extraction from images and video streams, the *definition of similarity measures* that can be applied to approximate human similarity judgment and new, better *concepts for the user interface* aspect of visual information retrieval, particularly for human-machine interaction for query definition and refinement and video handling. On top of this framework *working prototypes* are implemented that are fully based on the visual part of the *MPEG-7 standard* for multimedia content description. Reaching this goal requires the careful design of the database structure and an extendible class framework as well as research on suitable extensions and *supplementations* of the MPEG-7 standard by additional descriptors and descriptor schemes. Mathematical and logical fitting distance measures have to be selected for all descriptors (distance measures are not defined in the standard) and an appropriate and flexible *model for similarity definition* has to be defined. MPEG-7 is not information retrieval specific. One goal

of this project is to apply the definitions of the standard to visual information retrieval problems.

Another goal is the development of a *general-purpose user interface framework* for visual information retrieval. This framework has to include a great variety of different properties: methods for query definition from examples or sketches, similarity definition by positioning of visual examples in a 3D space, appropriate result display and refinement techniques and cognitively easy handling of visual content, especially video. User interfaces and querying methods both have to support methods for *distributed querying, storage and replication* of visual information and features as well as methods for query acceleration. The importance of this issue becomes apparent from the large amount of data that has to be handled and the computation power that is necessary for querying by—often quite complex—distance functions. Methods for distributed querying, storage and replication include the replication of feature information, client-server architectures and remote method invocation in the querying and indexing modules as well as compression of video representations for the transport over low bandwidth networks. Methods for query acceleration include indexing schemes, mathematical methods for complexity reduction of distance functions and the generation of querying heuristics [27].

An additional, however, implicit goal of the VizIR project is the development of a *multimedia-specific UML-based software development process*. Multimedia applications have special needs that have to be considered during the system design and implementation. This includes modeling of real-time media processing (multiplexing, conversion with codecs, rendering, etc.), more sophisticated modeling of users and use-cases (e.g. abstraction of users to user profiles, etc.), metadata modeling and modeling of multimedia restrictions (Quality of Service parameters, interaction, etc.). Developing tailor-made software development methods on the basis of the UML design process is just a natural consequence.

4. VizIR framework design

This section describes technical details of the VizIR objectives and the intended system architecture. The VizIR framework can be split into four areas of work: (1) querying framework, (2) user interface framework, (3) configuration and communication interfaces and (4) assessment methods. The querying framework contains all methods for feature extraction, similarity measurement, query refinement, media handling and database access. The user interface framework contains a class hierarchy of user interface elements (panels), events and event handling methods and media visualization classes. Configuration and communication concerns all classes and methods for standardized communication of framework elements with other elements (e.g. query engines and user interfaces) or the environment. Assessment methods include benchmarking techniques and media sets for evaluation. The next four subsections detail the relevant design issues for these areas of work.

4.1. Querying framework

The most important issue related to the design and implementation of the querying framework is the implementation of a technically sound class framework for the system components. Even though this is not a research but a software engineering problem, we have to stress that using a professional database and programming environment are crucial success factors for a modern VIR research prototype. As pointed out above, most past approaches have serious shortages in their system architecture.

VizIR uses a relational database for media and feature data storage. Fig. 1 gives an overview of its data model and indicates the relationships between media and feature storage. Visual media is stored in table *Media* and associated with a single *MediaType*. Each media may belong to n collections and each collection may contain m elements. Descriptors are described in table *FeatureClass* with the MPEG-7 descriptor definition language (DDL; based on XML schema). Feature data for a certain descriptor is stored in binary and/or XML format in table *FeatureData*. To allow the implementation of MPEG-7 descriptor schemes, descriptors are organized in collections in table *FeatureCollection*. A collection may consist of descriptors and other collections. Optionally, it may have a DDL-description. Based on this data

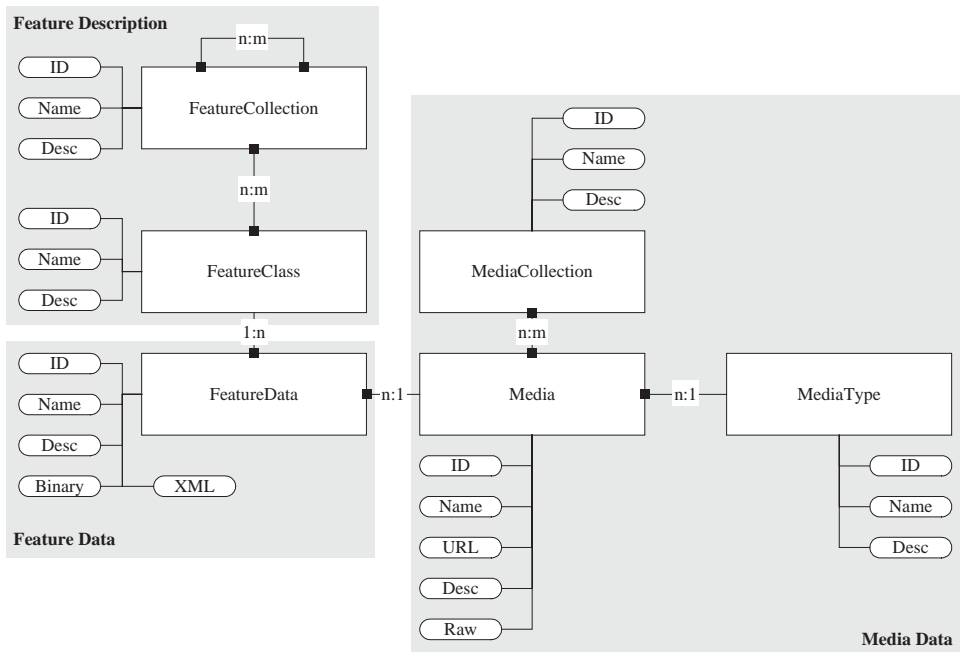


Fig. 1. EER database diagram. The framework contains a database manager that creates this structure during VizIR installation automatically.

model it is possible to use descriptor schemes in queries. If a certain feature collection is selected for a query, all referenced descriptors are selected and used in the query.

Fig. 2 outlines the class structure of the querying framework. To a certain extent this class framework follows the architecture of IBM's QBIC system [9], but largely differs from QBIC in its server/client independent classes. Similarly to QBIC, the database access is hidden from the feature programmer and the structure of all feature classes is predefined by an interface. Key element is class *QueryEngine*, which contains the methods for query generation and execution. Each query consists of a number of *QueryLayer* elements each of which implement exactly one feature. The result of each query is a set of media objects that is stored in a *Vector* object. Media objects are represented by objects of class *MediaContent*. *MediaContent* has an interface that hides the complexity of the actual media access from the framework programmer. For example, he can access the media data—independent whether it is image or video—with a method *getViewAtTime(Time, ColorModel)*. For images, *Time* is irrelevant and for videos it is the position in the media stream. The *ColorModel* of the resulting image can be RGB, HMMD, etc. With the *MediaContent*-mechanism CBIR and CBVIR can be implemented in the same framework without having to introduce media-specific peculiarities in the architecture. Similarly, the methods for database access are encapsulated in the *DatabaseManager*.

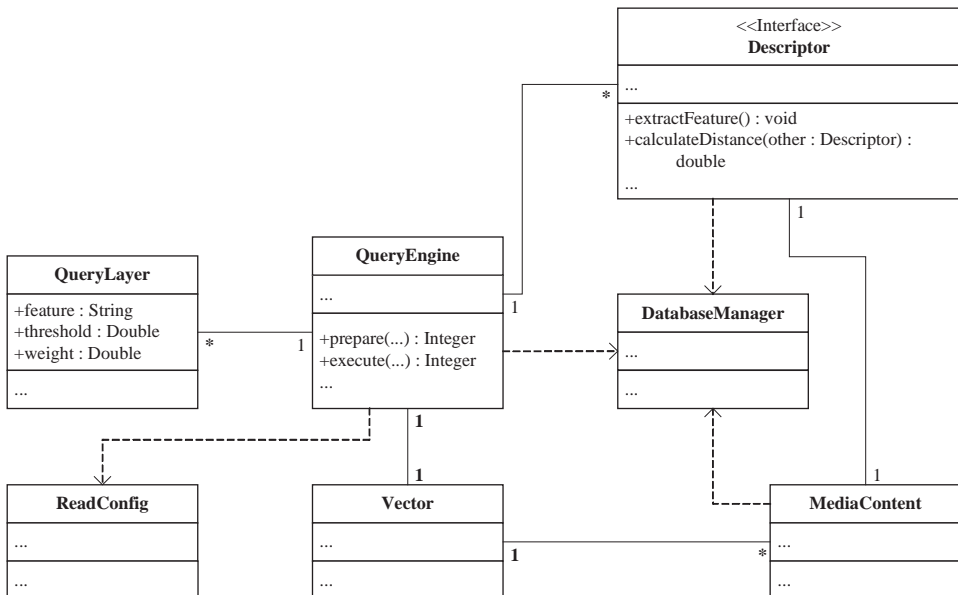


Fig. 2. UML class diagram for an ideal implementation of the VizIR class framework. Custom query engines can be added by sub-classing *QueryEngine*. The *DatabaseManager* offers a standard interface for accessing arbitrary relational databases. Similar to that, *MediaContent* offers methods for media access that hide the actually used media processing library.

All feature classes—MPEG-7 descriptors as well as all others—are derived from the interface *Descriptor*. For the MPEG-7 descriptors it is intended to follow the reference implementation of part 6 of the standard. For the reasons given above and especially, because the algorithms of the reference implementation are not performance-optimized the redesign and implementation of the MPEG-7 descriptors is a time- and human resources-consuming task.

Descriptor contains methods for descriptor extraction (*extractFeature()*) and distance measurement (*calculateDistance()*). Unfortunately (for us), MPEG-7 is not a visual information retrieval-specific standard and in general does not include distance functions for the various descriptors. Neither does it give any recommendations for their selection. Therefore it is necessary to implement common distance metrics (like L^1 -, L^2 -metric, Mahalanobis distance, etc.; [2]), to associate them with descriptors and to find custom distance functions where these metrics are not applicable (e.g. object features, etc.).

The *extractFeature()*-method of *Descriptor* applies the actual feature extraction algorithm to the media considered (and accessible) as *MediaContent*. The MPEG-7 standard—although it is a major advance in multimedia content description—standardizes a number but not all useful features. It is necessary to implement additional descriptors and distance functions for texture description of images (wavelets, etc.; e.g. [28]), symmetry detection of objects (useful for face detection, detection of human-made objects, etc.), object description in video streams (structure recognition from motion, etc.), object representation (scene graphs, etc.) and video analysis (shot detection, etc.). Additionally, we plan to use fractal methods (iterated function systems; IFS) to describe the shape of objects effectively. So far IFS have been used for the compression of self-similar objects (e.g. [29]) but hardly for content-based retrieval (see [30]). We think, that IFS could be very effective for shape description, too.

The sequence diagram in Fig. 3 depicts the querying process. Methods for query definition and query refinement have to be flexible enough to satisfy different ways of how humans perceive and judge similarity and should still be applicable in a distributed querying environment. In VizIR each type of application (server, Servlet, client, applet, etc.) can initiate a query by instantiating a *QueryEngine* object and calling the *prepare()* method. The *execute()* method of a query creates a feature class for each *QueryLayer* of a query and extracts a descriptor by calling *extractFeature()*. These objects of class *Descriptor* are then used for feature comparison with *Descriptor* objects of the images in the database by the method *calculateDistance()*. The images of the result set are returned via the *getElements()* method. To accelerate queries, indexing schemes and other query acceleration models will be implemented as part of VizIR. Next to classic index structures for visual content (e.g. R-tree, segment index tree, etc.) and query acceleration techniques (storage of the factorized terms of the Mahalanobis distance [31], etc.), experiments will be undertaken with new heuristic approaches like those we previously published [27].

Concluding this sketch of the VizIR querying framework architecture we outline several aspects of the application and data distribution. In a scalable framework it is simply necessary to implement tools for distributed and replicated visual content

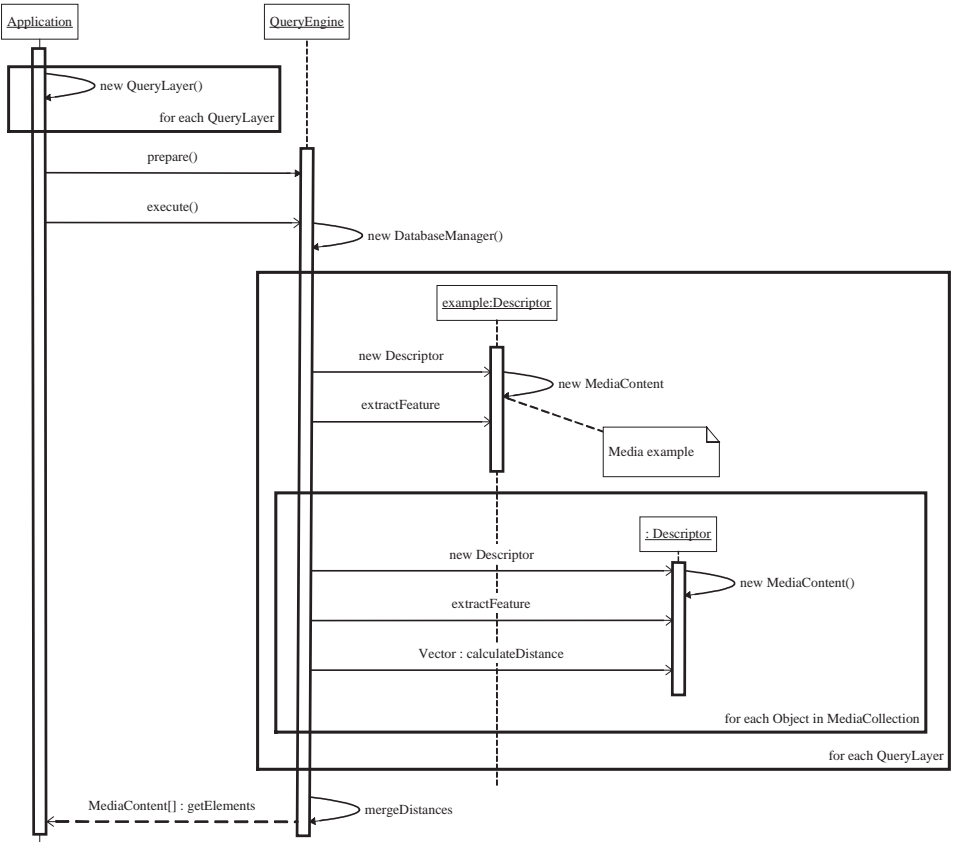


Fig. 3. Schematic UML sequence diagram of the querying process.

management as well as database management. Modern Web Service- or CORBA-based programming environments like the Java environment permit the network-independent distribution of applications, objects and methods (in Java through the Remote Method Invocation library) to increase the performance of an application by load balancing and multi-threading. VizIR is based on Java. Therefore the objects for querying can be implemented as JavaBeans, feature extraction functions with RMI, database management through Servlets and user interfaces as Applets. Database distribution is realized through standard replication mechanisms and database access through JDBC.

4.2. User interface framework

The VizIR user interface framework is a collection of components that can be combined arbitrarily. The major issue is the design of querying & query refinement interfaces that integrate image and video content, the implementation of methods for

video content representation in static user interfaces and the support of multiple media-based querying paradigms. All user interface components have to be designed as intuitive and self-explanatory as possible to guarantee high usability and, as a consequence, increasing acceptance of VIR. In addition to user interface building blocks, methods have to be developed that allow their combination in application-specific user interfaces (fields of application in the future will be digital libraries, medical image search, TV broadcast archives, etc.).

Fig. 4 shows the static structure of the VizIR user interface framework that should satisfy these demands. Central element is the interface *UserInterfaceComponent* that is inherited by all classes having a visual panel. These are *MediaPanel* (the mother class of all panels that deal with media objects), *QueryEngine* (the mother class of all querying engines, the panel contains all elements necessary for query formulation), *Descriptor* (mother class of all implemented features, the panel contains a toolbox for sketch drawing), *MetadataPanel* and *LayerPanel* (a layer manager for multi-layer

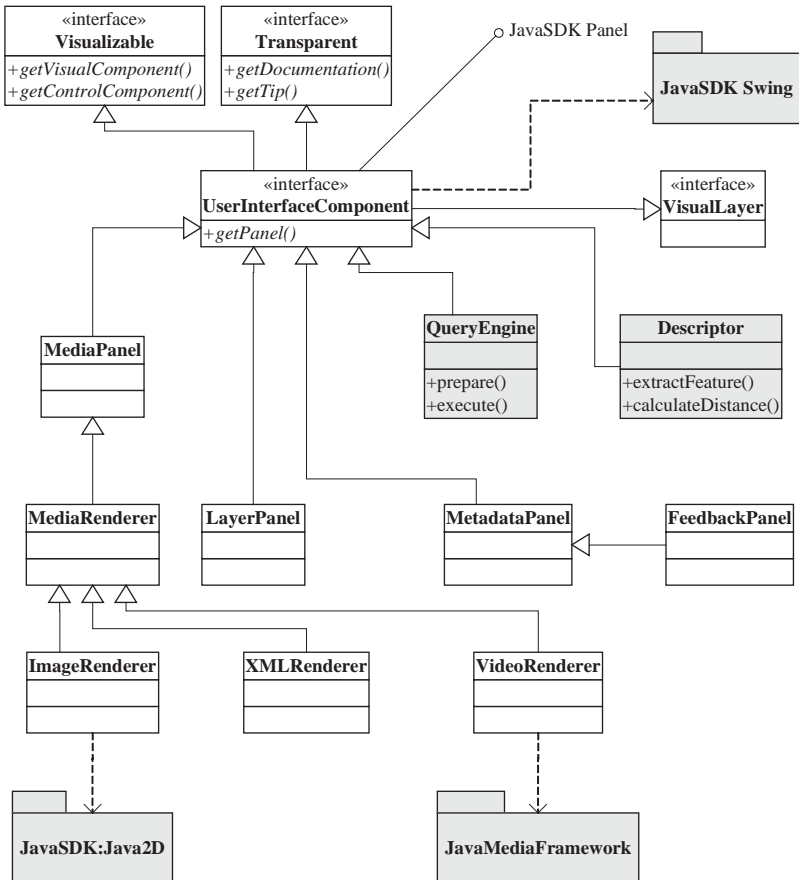


Fig. 4. Class diagram of the VizIR user interface framework.

image sketching as in Photoshop). VIZIR user interface components are based on Java Swing. *UserInterfaceComponent* inherits methods from the interfaces *Visualizable* (methods for receiving a visual panel and a visual control component like in the Java Media Framework [32]), *Transparent* (methods for receiving visual documentation and help in the user interface) and *VisualLayer* (defines the structure of a layer of the sketching panel, basically a Java *Image* type).

MediaRenderer is a special type of *MediaPanel* for the visual rendering of media objects. *MediaRenderer* takes an arbitrary media object as input and generates a (2D or 3D) diagrammatic representation. Representing media objects in a static user interface is easy for images but difficult for (time-based) video content. Common approaches are index frames and Micons, which obviously are unsatisfactory. A more sophisticated approach would be an object viewer for all objects and their temporal trajectories in a video shot. Also, video cubism (allowing for interactively cutting an X – Y –time cube of video data along arbitrary oriented planes; [33]) should be considered as an alternative for presenting video results. So far, we have implemented three renderers for images (JPG, PNG, GIF, etc., based on Java2D), videos (generates Micons—see Section 2.2—for arbitrary video formats: MPG, AVI, MOV, etc., based on the Java Media Framework) and XML. *XMLRenderer* can render any XML-file that can be displayed in a web browser (see [7] for technical details). Fig. 5 shows examples: element A and B are Micons (representing videos of

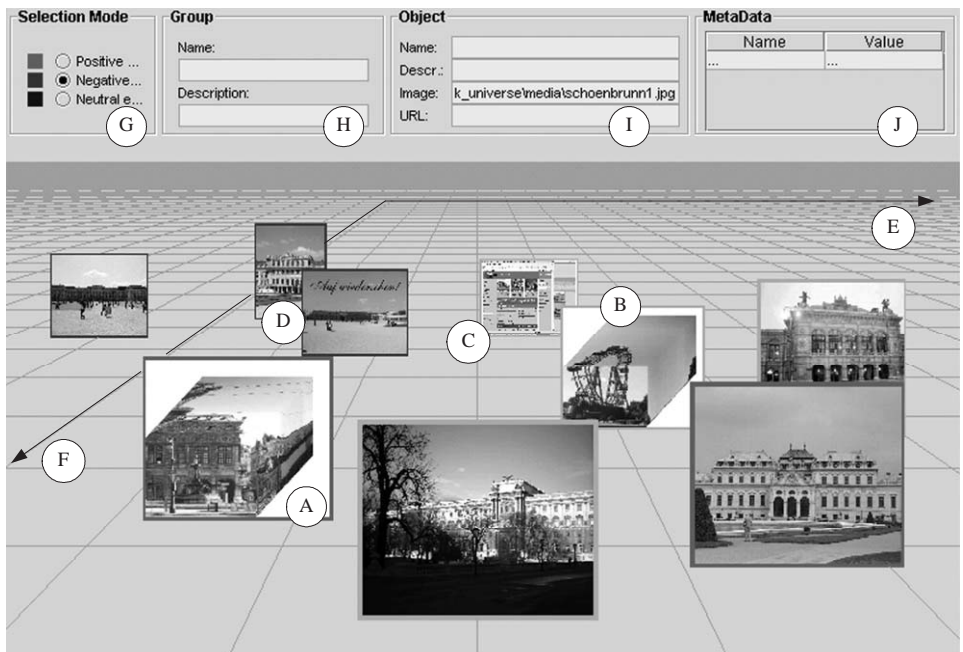


Fig. 5. Screenshot of the 2.5D panel (media objects are positioned at random).

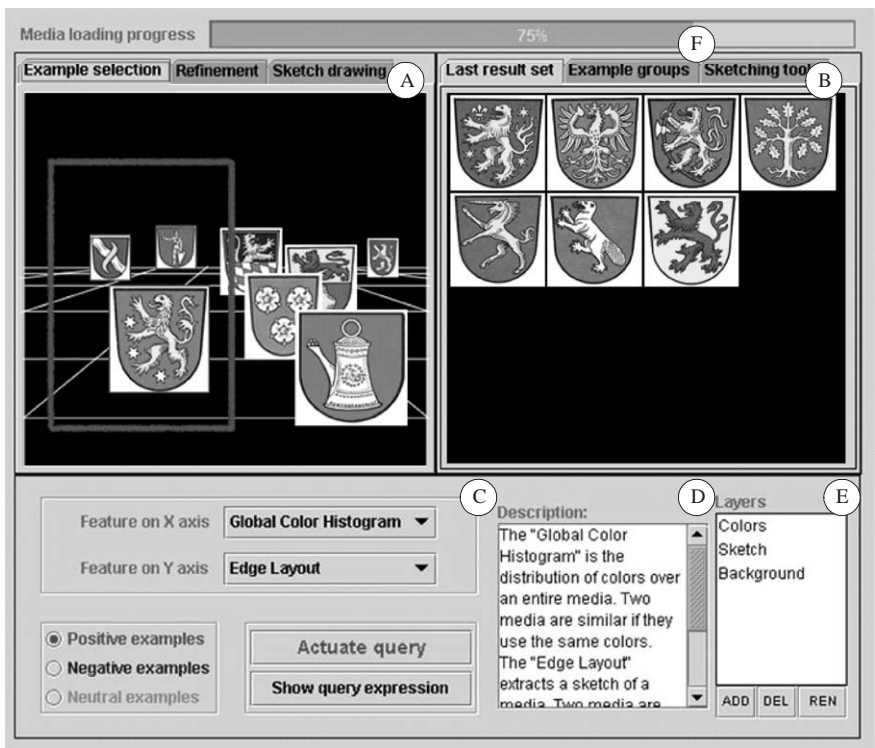


Fig. 6. Screenshot of a VizIR user interface prototype.

the Vienna opera house and the Prater Ferris wheel), element C is a webpage and all other media objects are standardized images. Like *MediaContent* for media access, the *MediaRenderer*-mechanism allows performing CBIR and CBVR in the same user interfaces and the implementation of unified APIs for both types of media.

The most important *MediaPanel* is the 2.5D media panel. For examples see Fig. 5 and element A of Fig. 6. The 2.5D panel is used for example selection, browsing, query formulation and the display of result sets. The rendered substitutes of media objects are displayed as images parallel to the image plane. It is possible to navigate in two dimensions (left–right, forward–back) and to zoom. Groups of objects can be selected, moved and associated with metadata (by communication with a *MetadataPanel*). The angle of the image plane and the X – Y -plane can be varied between 0° and 90° . The panel may have visual control components (elements G–J in Fig. 5 and element C in Fig. 6). Panel G in Fig. 5 (also shown in the lower left part of element C in Fig. 6) allows to set the selection mode for the cursor and panel H is for group definition. Panel I shows information on the currently selected object and panel J its metadata entries. The upper panel of element C of Fig. 6 is initialized with all dimensions of the media space to be displayed (in the VIR context: all implemented features). The view changes whenever new dimensions are chosen for

the X - or Y -axis or the querying button in the lower right part of element C in Fig. 6 is pressed.

It is important to know—in rough terms—the querying process implemented in VizIR to understand the role of the 2.5D panel. Fig. 7 shows a State-Transition-Diagram of the underlying querying process. First the user interface components are initialized with media objects and query parameters (element F of Fig. 6 shows a progress bar panel for media loading). Then the user can define a first query by selecting example media objects. This sets the user interface in the defined state. Executing the query brings the user interface in the active state where refinement can be started or a new query can be defined. In active state the query is re-executed whenever the user presses the ‘activate’ button or the query engine control component detects substantial changes in the query definition.

Both panels for query definition and query refinement are 2.5D panels that have been initialized with MRML-documents. They can visualize any two-dimensional subspace of the distance space (for the selected features and examples) generated in the previous querying iteration. This is done by showing the media objects (or their representations) parallel to the image plane and, on the X - and Y -axis, arranged according to their relative distance (depicted in element E and F in Fig. 5). Similar objects are placed near to each other, un-similar objects far from each other. Element A of Fig. 6 shows the distance of images for a color histogram feature on the X -axis and the distance for an edge histogram on the Y -axis. The features (distance space dimensions) shown on the X - and Y -axis can be changed *interactively*. Queries are defined and refined in the same way by selecting media objects or groups and

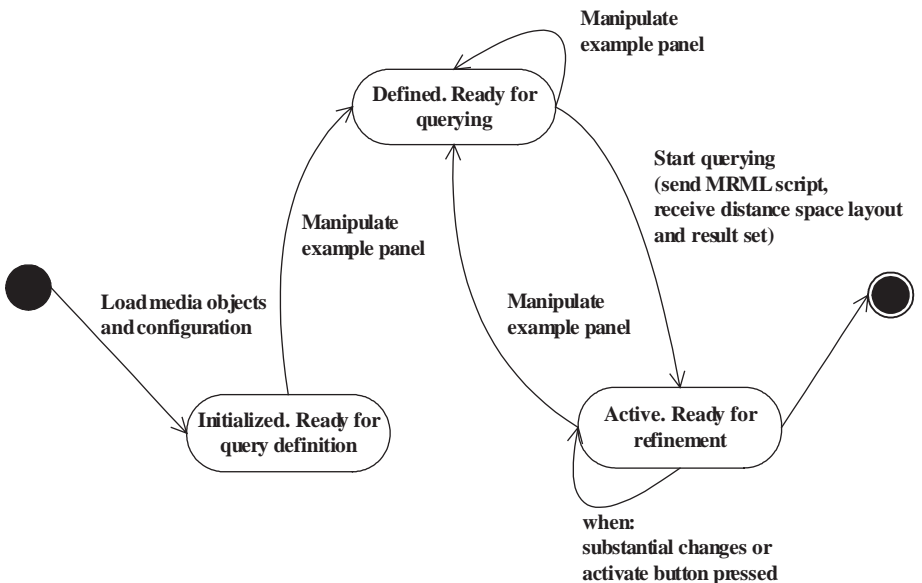


Fig. 7. State-transition-diagram of the querying process.

marking them as positive or negative examples. Thus, it is possible to define n -dimensional hyper-cubes (clusters) of (un-)similar media objects. The query engine tries to find all media objects that belong to the clusters with positive examples minus those with negative examples. We call this similarity measurement process Logical Retrieval (LR, see [35] for a more detailed description).

We are implementing two querying paradigms: query by example (QBE) and query by sketch (QBS), because they are media-based and intuitive. Even though for beginners text querying may be the easiest form of interaction, we are—at this point in time—not planning to implement a text interface, because implementing such an interface would not raise new VIR research questions nor help to solve the existing ones. QBE follows the querying process described above. Sketches for QBS can be drawn in the ‘sketch drawing’ panel in Fig. 6. This panel contains layers of type *VisualLayer* that are managed by the *LayerManager* (element E in Fig. 6) and allow drawing with the tools provided by the descriptor objects. These tools are collected in the ‘sketching tools’ panel (element B in Fig. 6). The ‘last result set’ panel contains the media objects of the last result set (similarity values are associated as metadata). It is just a special 2.5D example panel with an image-plane to X – Y -plane angle of 0° . The same is true for the ‘example groups’ panel in Fig. 6 that lists all query examples partitioned in three groups: positive, negative and neutral examples. (Neutral examples are explicitly excluded from the query. Their properties are marked as irrelevant for the query.) The ‘description’ panel (element D in Fig. 6) contains the information of the methods from the *Transparent* interface for the active user interface element.

The VizIR user interface class structure follows the paradigm that all components (methods, panels, etc.) are defined, where they are used. Thus, each query engine has a visual panel for query formulation and each descriptor has a panel with tools for sketching (e.g. line drawing tools for an edge layout descriptor). To guarantee the transparency of VizIR (defined in [6]), each visual component has to implement the *Transparent* interface with documentation and tips. The panels of the framework can be integrated into any visual Java container and organized arbitrarily. The layouts in the screenshots in Figs. 5 and 6 are just examples. Because the VizIR framework is based on Java and the Java SDK is possible to integrate the user interface components into any container (frame, applet, etc.) to perform distributed querying (with Web Services, CORBA, RMI, etc.) and querying in the background (in a separate thread).

The validity of arbitrary combinations is guaranteed by the communication mechanism of the framework. It follows the Delegation-Event-Model and is conceptually shown in Fig. 8. Each object of class *MediaPanel* (*MediaPanel-1* and *MediaPanel-2*) may communicate with any other *MediaPanel* through *MediaPanel-Event* objects (e.g. the selection mode panel in element G in Fig. 5 with a 2.5D panel). Thus, all media panels have to implement listener classes that are defined in *UserInterfaceComponent* and flag the media panel events they fire. For easier user interface building the framework contains convenience classes with listener functions for standard communication operations (e.g. communication of query control panel and 2.5D panel when the example group selection is changed, etc.).

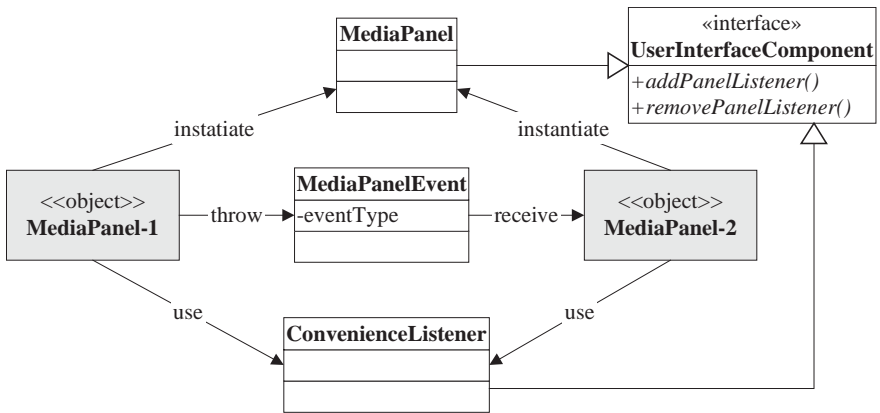


Fig. 8. Event model for panel communication. Media panels communicate through *MediaPanelEvent* objects.

4.3. Configuration and communication interfaces

Query engines in VizIR can be of arbitrary kind. We are implementing a query engine based on the querying process sketched in the previous section (see [35] for more details). In VizIR, the communication of user interfaces and query engines is loosely coupled based on MRML (see Section 2.2).

Each framework component that uses MRML for communication, uses instances of the classes *MRMLReader* and *MRMLWriter* (see Fig. 9). These classes are derived from *ReadConfig* (XML parser class) and *WriteConfig* (XML writer class). Communication classes for new XML languages can be implemented in the same way. In order to perform LR queries with MRML we had to extend its document type definition (see Appendix A for DTD code). We have defined elements for context-free media and media group definition (required for the implemented querying paradigm), descriptor definition and query definition. The following example illustrates how these extensions can be used:

```

<logicalQuery>
  <clusterDefinition>
    <clusterRestriction>
      <clusterDimension lowerBound='0.0'
        upperBound='0.5'>
        <mediaGroup id='qe1' type='positive'>
          <mediaObject dataLocation='file:img1.gif'
            iconLocation='file:thumb1.gif' />
        </mediaGroup>
        <descriptor name='ColorHistogram'>
      </clusterDimension>
    </clusterRestriction>
  </clusterDefinition>
</logicalQuery>
  
```

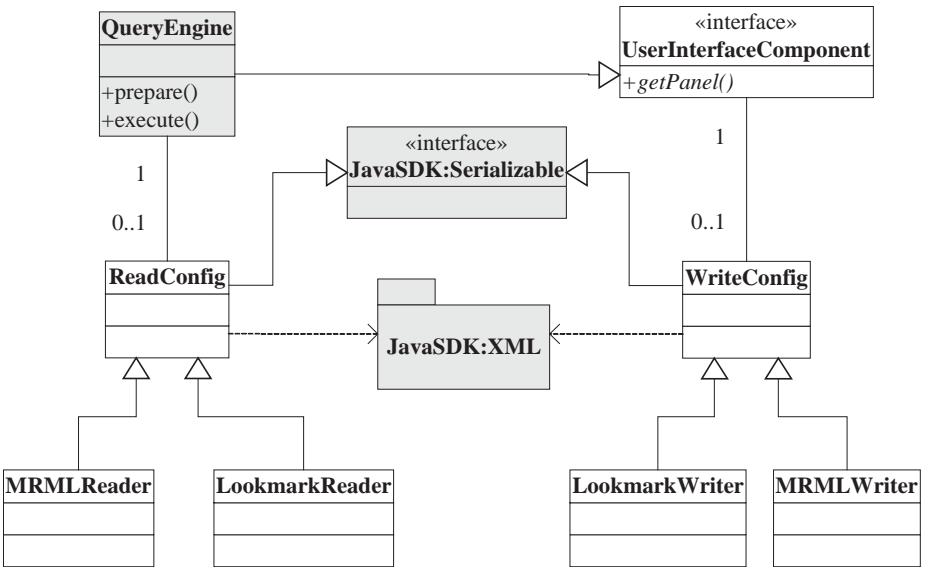


Fig. 9. Class diagram for MRML communication in VizIR. Query engines and user interface components use the classes *ReadConfig* and *WriteConfig* or their subclasses to read and write XML configuration files.

```

</clusterRestriction>
</clusterDefinition>
</logicalQuery>

```

This construct defines a query (on the *collection* defined elsewhere in the MRML script) with a single feature. A color histogram is used to find all media objects that have a distance to the positive query example ‘img1.gif’ (represented by the icon ‘thumb1.gif’) that is smaller than ‘0.5’. If we liked to retrieve all objects that fulfil this cluster-condition *and* a second one, we would put the second *clusterRestriction* in the same *clusterDefinition*. If we wanted to retrieve all media objects meeting the first *or* the second condition, we would put the second one in a new *clusterDefinition*. These constructs are flexible and can be used in various ways. They should not only support our LR concept but—according to the published querying paradigm—the one used in MARS as well [13].

4.4. Assessment methods

Concluding this description of the VizIR framework, we would like to point out issues that are related to VIR assessment methods. To our belief, a significant improvement of VIR research in the future will be the development of standardized quality assessment procedures (like in the Benchathlon project [4]). In the VizIR

project the following assessment tasks will be undertaken:

- Formulation of standardized evaluation procedures.
- Collection and creation of media sets with ground truth.
- Evaluation of descriptors and querying methods.
- Evaluation of query acceleration methods.

Common evaluation models (recall, precision, etc. [36,37]) are analyzed to develop standardized evaluation procedures. The application of the standard measures in information retrieval, recall and precision, to VIR systems using linear weighted merging (see above) implies giving up at least 10% of recall, since a system with linear weighted merging returns the n ‘most similar’ available objects (independent of the question whether or not they are really similar), while the recall measures the ratio of really similar objects to all available objects. This has to be considered in the evaluation process. As a consequence, the feasibility of less well-known methods (systematic measures, etc.) will be investigated and methods from other research areas will be checked for applicability. This could be psychological methods, e.g. semantic differential techniques [38].

Evaluation sets with image and video content will be collected and—where not available—created for groups of descriptors and ground truth information will be derived from tests with volunteers (students, etc.). Such sets are obviously decisive for the quality judgment of VIR systems. Actually, however, only a few de-facto standards do exist, including the Brodatz database for texture images. Partially, these evaluation sets will be created by enriching and extending the image and video clip sets, used for the MPEG-7 evaluation. As well, different approaches—e.g. findings on the basis of gestalt laws—will be checked for their suitability to develop those test sets.

The extended evaluation of the MPEG-7 descriptors, descriptor schemes and other implemented descriptors with statistical methods will be performed in two steps: (1) Evaluation of their independent performance and their performance in combinations. From this information the overall performance of the visual part of MPEG-7 and VizIR can be judged. (2) Analysis of dependencies among descriptors with statistical methods (cluster analysis, factor analysis, etc.) to identify a base for the space of descriptors and to be able to normalize the visual part of the MPEG-7 standard and to extend it by new independent descriptors.

Finally, the performance optimization methods developed for VizIR will be compared to those developed for other comparable retrieval systems. In the past, we have implemented several performance optimization techniques and compared them by the reply time for queries (e.g. in [39]). This will be continued in VizIR.

5. Implementation

In this section, two major implementation decisions of VizIR are discussed: the choice of the programming environments for media handling and graphic i/o. When we made these decisions, we had not yet decided if we should base VizIR on C++ or Java.

5.1. Media programming environment

The major question concerning the implementation of the VizIR prototype is the programming environment. At this point in time, there are three major alternatives that support image and video processing to choose from:

- Java and the Java Media Framework (JMF; [32]).
- The Open Media Library standard (OpenML) of the Khronos group [40].
- Microsoft DirectX (namely DirectShow [41]).

All of these environments offer comprehensive video processing capabilities and are based on modern, object-oriented programming paradigms. DirectX is limited to Windows-operating systems and a commercial product. Therefore, in the following discussion we will concentrate on the first two alternatives: JMF and OpenML. JMF is a platform-dependent add-on to the Java SDK, which is currently available for SunOS, Windows, MacOS-X (implementation by SUN and IBM) as well as Linux (implementation by Blackdown) in a full version and in a Java version with less features for all other operating systems that have Java Virtual Machine implementations. JMF is free and extensible. OpenML is an initiative of the Khronos Group (a consortium of companies with expert knowledge in video processing, including Intel, SGI and SUN) that standardizes a C-interface for multimedia programming. OpenML includes OpenGL for 3D and 2D vector graphics, extensions to OpenGL for synchronization, the MLdc library for video and audio rendering and the ‘OpenML core’ for media processing (unfortunately, the media processing part of OpenML is named OpenML as well; therefore we will use the term ‘OpenML-mp’ for the media processing capabilities below). Lately, the first implementation of the OpenML SDK was announced for summer 2003 (for Irix).

Among the concepts that are implemented in a similar fashion in JMF and OpenML-mp are the following:

- Synchronization: a media object’s time base (JMF: *TimeBase* object, OpenML-mp: Media Stream Counter) is derived from a single global time base (JMF: *SystemTimeBase* object, OpenML-mp: Unadjusted System Time).
- Streaming: both environments do not manipulate media data as a continuous stream, but instead as discrete segments in buffer elements.
- Processing control: JMF uses *Control* objects and OpenML-mp uses messages for this purpose.

Other important media processing concepts are implemented differently in JMF and OpenML-mp:

- Processing chains: in JMF real-time processing chains with parallel processing can be defined (one instance for one media track is called a Codec Chain). In OpenML-mp processing operations data always flow from the application to a single processor (called a Transcoder) through a pipe and back.

- Data flow: JMF distinguishes between data sources (including capture devices, RTP servers and files) and data sinks. OpenML-mp handles all I/O devices in the same way (called Jacks).

The major advantages of OpenML-mp are:

- Integration of OpenGL, the platform-independent open standard for 3D graphics.
- A low-level C API that will probably be supported by the decisive video hardware manufacturers and should have a superior processing performance.
- The rendering engine of OpenML (MLdc) seems to have a more elaborate design than the JMF renderer components. Especially, it can be expected that the genlocking-mechanism of MLdc will prevent lost-sync phenomena, usually occurring in JMF when rendering media content with audio and video tracks longer than 10 minutes.
- OpenML-mp defines more parameters for video formats and is closer related to professional video formats (DV, DVCPRO, D1, etc.) and television formats (NSTC, PAL, HDTV, etc.)

On the other hand the major disadvantages of OpenML are:

- It is not embedded in a CASE environment like Java for JMF. Therefore application development requires more resources and longer development cycles.
- OpenML is not object-oriented and does not include a mechanism for parallel media processing.

The major drawbacks of JMF are:

- Lower processing performance because of the high-level architecture of the Java Virtual Machine. This can be reduced by the integration of native C code with the Java Native Interface.
- Limited video hardware and video format support: JMF has problems with accessing certain video codecs, capture devices and with transcoding of some video formats.

The outstanding features of JMF are:

- Full Java integration. The Java SDK includes powerful methods for distributed and parallel programming, database access and I/O processing. Additionally, professional CASE tools exist for software engineering with Java.
- JMF is free software and reference implementations exist for a number of operating systems. JMF version 2.0 is a co-production of SUN and IBM. In version 1.0, Intel was involved as well.
- JMF is extensible. Additional codecs, multiplexers and other components can be added by the application programmer.

The major demands for the VizIR project are the need for a free and bug-free media processing environment that supports distributed software engineering and has a distinct and robust structure. Issues as processing performance and extended

hardware support are secondary for the project. Therefore we think JMF currently being the best choice for the implementation.

Design and implementation follow an UML-based incremental design process and rely on prototyping. UML and prototyping are employed, because they both represent state-of-the-art in software engineering. Prototyping, in addition, shows invaluable positive effect on the motivation of the developers.

5.2. User interface and communication issues

One of the most important elements of the user interface class framework is the 2.5D panel. It is based on G14Java [34] instead of Java3D for the following reasons: (1) G14Java is based on OpenGL and much faster than Java3D, (2) event handling is easier and bug-free, (3) it is easier to install (e.g. less dependent on graphics hardware than Java3D) and (4) has less bugs than Java3D.

XML reader and writer classes are based on the Java XML package (JAXP). We use the JDOM parser for XML writing (because it allows the construction of an object tree in memory and does serialization automatically) and SAX for XML parsing (because it is more flexible and faster than JDOM).

A special communication problem of VIR user interfaces is the transportation of media objects to the client computer. We do media loading in the background through an RTP stream. The Java Media Framework contains a convenient RTP-based streaming component. The user interface is operational as soon as at least a certain quantity of the media objects has arrived at the client side. This is improved by first sending a subset of representative media objects through the stream.

6. Past, current and future work

Currently, we are working on the first release of the VizIR framework. Most components of the querying framework, the database manager, the basic user interface framework (including a video renderer and a webpage renderer for thumbnail creation), the 2.5D panel and the XML communication classes are finished since autumn 2002. Next, we will implement a general-purpose query engine, a unified media handler for images and video and some of the MPEG-7 visual descriptors. A first prototype of the full framework should be finished by autumn 2003. This first version (and all following) will be released under GNU Public License.

Next we will work on other methods for feature extraction, distance measurement and video representation. New feature extraction methods we are currently working on, are semantic feature classes that enrich existing descriptor data of low-level features (e.g. MPEG-7 descriptors) with additional knowledge (modeling information, statistical dependencies, etc.) to reduce the impact of the semantic gap (first results in [42]). Concerning video representation, we will follow two approaches. First, we will implement a renderer that produces animated icons of selected keyframes of a video. The keyframes will indicate scene changes. The second

approach originates in 2D animation. Short sequences of keyframes will be overlaid with an alpha-channel and thus integrated into a video thumbnail. Another idea that we will follow in the future, is the implicit definition of features from the selection of media elements or media element regions and expert knowledge. In the past we have been working on a similar idea that resulted in the system presented in [20].

7. Conclusion

This paper describes the querying and user interface framework of the Visual Information Retrieval project VizIR. The framework consists of a class hierarchy of querying classes and user interface panels with event communication, communication and configuration methods based on XML and an extension of the MRML for communication of user interfaces and query engines. The intended major outcome of the VizIR project can be summarized as follows:

- An open class framework of methods for feature extraction, distance calculation, user interface components and querying.
- Evaluated user interface components and prototypes for content-based visual retrieval.
- System prototypes for the refinement of the basic methods and interface paradigms.
- Carefully selected evaluation sets for groups of features (color, texture, shape, motion, etc.) with human-rated co-similarity values.
- Evaluation results for the methods of the MPEG-7 standard, our earlier content-based retrieval projects and other promising methods.

VizIR is open, extendible and free. A first version of the user interface part (3D interaction panel, XML-communication classes) is available since autumn 2002, the first release of the full framework should be ready by autumn 2003 and will be available under GNU Public License. We would like to invite interested research institutions to join the discussion and participate in the design and implementation of the open VizIR project. Contact the authors to join the project and/or get a copy of the available pre-release software.

Acknowledgements

The VizIR project is supported by the Austrian Scientific Research Fund (FWF) under grant no. P16111–No 5.

Appendix A

This appendix contains the document type definition (DTD) for the essential part of our MRML extension. The extension includes elements for context-free media

and media group definition, descriptor definition and query definition according to our querying paradigm. It is based on version 1.0 of the MRML definition presented in [8] (see Section 2.2 for details). The tags below can be easily integrated into MRML by adding *logicalQuery* and *mediaGroup* as sub-tags of the *mrml* tag.

A.1. Media and media group definition

In MRML media objects can be context-sensitively defined as *user-relevance-elements* (for querying) or as *query-result-elements*. For initialization we add a tag for general media definition:

```
<!ELEMENT mediaObject (descriptor*)>
<!ATTLIST mediaObject
  dataLocation CDATA #REQUIRED
  iconLocation CDATA #REQUIRED>
```

dataLocation and *iconLocation* are URLs. As far as we understand, the *collection* tag of MRML cannot be used for the definition of media groups (for querying, etc.). We define the following element for this purpose:

```
<!ELEMENT mediaGroup (mediaObject+)>
<!ATTLIST mediaGroup
  id CDATA #REQUIRED
  type (positive|negative|neutral|init|other) 'positive'>
```

The first three types define querying groups. The fourth is for initialization. Neutral examples are explicitly excluded from the query. Their properties are marked as irrelevant for the querying process.

A.2. Descriptor definition

MRML uses the *algorithm*-construct for the definition of features. For extended use we define arbitrary descriptors as follows:

```
<!ELEMENT descriptor EMPTY>
<!ATTLIST descriptor
  name CDATA #REQUIRED
  value CDATA
  distanceValue CDATA>
```

distanceValue is a special field used only when media objects are grouped to describe the layout in distance space (related to the query examples) instead of feature space.

A.3. Logical retrieval query definition

According to our Logical Retrieval approach, a query can be defined by the following elements:

```
<!ELEMENT logicalQuery (clusterDefinition+)>
<!ELEMENT clusterDefinition (clusterRestriction+)>
<!ELEMENT clusterRestriction (clusterDimension+)>
<!ELEMENT clusterDimension (mediaGroup,descriptor)>
<!ATTLIST clusterDimension
    lowerBound CDATA #REQUIRED
    upperBound CDATA #REQUIRED>
```

See Section 4.3 for an example.

References

- [1] S. Santini, R. Jain, Beyond query by example. *ACM Multimedia*, (1998) 345–350.
- [2] S. Santini, R. Jain, Similarity Measures, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21 (1999) 871–883.
- [3] J. Barros, J. French, W. Martin, Using the triangle inequality to reduce the number of comparisons required for similarity based retrieval, in: *Proceedings of SPIE Storage and Retrieval for Image and Video Databases*, San Jose, USA, 1996, pp. 392–403.
- [4] Benchathlon Network Website. <http://www.benchathlon.net> (last visited: 2003–03–20).
- [5] TREC video retrieval competition website. <http://www-nlpir.nist.gov/projects/trecvid/> (last visited: 2003–03–20).
- [6] H. Eidenberger, C. Breiteneder A Framework for Visual Information Retrieval, in: *Proceedings of Visual Information Systems Conference, HSinChu, Taiwan*, 2002, pp. 105–116.
- [7] H. Eidenberger, C. Breiteneder A Framework for user interfaced design in Visual Information Retrieval, in: *Proceedings of IEEE Multimedia Software Engineering Symposium, Newport Beach, USA*, 2002, pp. 255–262 (published on CD).
- [8] Multimedia Retrieval Markup Language Website. <http://www.mrml.net> (last visited: 2003–03–20).
- [9] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, P. Yanker, Query by image and video content: the QBIC system, *IEEE Computer* 28 (9) (1995) 23–31.
- [10] J. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain, C. Shu, The Virage image search engine: an open framework for image management, in: *Proceedings of SPIE Storage and Retrieval for Image and Video Databases*, San Jose, USA, 1996, pp. 76–87.
- [11] J.R. Smith, S. Chang, VisualSEEK: a fully automated content-based image query system, in: *Proceedings of ACM Multimedia Conference*, Boston, USA, 1996, pp. 87–98.
- [12] A. Pentland, R.W. Picard, S. Sclaroff, Photobook: tools for content-based manipulation of Image databases, in: *Proceedings of SPIE Storage & Retrieval Image & Video Databases*, San Jose, USA, 1994, pp. 34–47.
- [13] M. Ortega, R. Yong, K. Chakrabarti, K. Porkaew, S. Mehrotra, T.S. Huang, Supporting ranked Boolean similarity queries in MARS, *IEEE Transactions on Knowledge and Data Engineering* 10 (6) (1998) 905–925.
- [14] S. Santini, R. Jain, Integrated browsing and querying for image databases, *IEEE Multimedia* 3 (7) (2000) 26–39.
- [15] GNU Image Finding Tool Website. <http://www.gnu.org/software/gif/> (last visited: 2003–03–20).

- [16] E. Oomoto, K. Tanaka, OVID: design and implementation of a video-object database system, *IEEE Transactions on Knowledge and Data Engineering* 5 (4) (1993) 629–643.
- [17] J.K. Wu, C. Lam, B.M. Mehre, Y.J. Gao, A. Desai Narasimhalu, Content-based retrieval for trademark registration, *Multimedia Tools and Applications* 3 (3) (1996) 245–267.
- [18] T. Chua, L. Ruan, AVideo retrieval and sequencing system, *ACM Transactions on Information Systems* 13 (4) (1995) 373–407.
- [19] MPEG-7 Documents Website. http://mpeg.telecomitalialab.com/working_documents.htm#MPEG-7 (last visited: 2003–03–20).
- [20] C. Breiteneder, H. Eidenberger, Automatic query generation for content-based image retrieval, in: *Proceedings of IEEE Multimedia Conference*, New York, USA, 2000, pp. 705–708.
- [21] G. Sheikholeslami, W. Chang, A. Zhang, Semantic clustering and querying on heterogeneous features for visual data, in: *Proceedings of ACM Multimedia Conference*, Bristol, UK, 1998, pp. 3–12.
- [22] M. Nakazato, L. Manola, T.S. Huang, ImageGrouper: Search, Annotate and organize images by groups, in: *Proceedings of Visual Information Systems Conference*, HsinChu, Taiwan, 2002, pp. 129–142.
- [23] M. Tavanti, M. Lind, 2D vs. 3D, implications on spatial memory, in: *Proceedings IEEE Symposium on Information Visualization*, San Diego, USA, 2001, pp. 139–145.
- [24] G. Robertson, M. Czerwinski, K. Larson, Data mountain: using spatial memory for document management, in: *Proceedings of ACM Symposium on User Interface Software and Technology*, San Francisco, USA, 1997, pp. 153–162.
- [25] D.A. Keim, Visual exploration of large data sets, *Communications of the ACM* 44 (8) (2001) 38–44.
- [26] B. Furht, S.W. Smoliar, H. Zhang, *Video and image processing in multimedia systems*, Kluwer Publishers, Boston, 1996.
- [27] C. Breiteneder, H. Eidenberger, Performance-optimized feature ordering for content-based image retrieval, in: *Proceedings of European Signal Processing Conference*, Tampere, Finland, 2000 (published on CD).
- [28] F. Liu, R.W. Picard, Periodicity, directionality, and randomness: wold features for image modeling and retrieval, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18 (7) (1996) 722–733.
- [29] M.F. Barnsley, L.P. Hurd, M.A. Gustavus, Fractal video compression, in: *Proceedings of IEEE Computer Society International Conference*, USA, 1992, pp. 41–42.
- [30] A. Lasfar, S. Mouline, D. Aboutajdine, H. Cherifi, Content-based retrieval in fractal coded image databases, in: *Proceedings of Visual Information and Information Systems Conference*, Amsterdam, Netherlands, 1999.
- [31] Y. Rui, T. Huang, S. Chang, Image retrieval: past, present and future, *Journal of Visual Communication and Image Representation* 10 (1997) 1–23.
- [32] Java Media Framework Website. <http://java.sun.com/products/java-media/jmf/> (last visited: 2003–03–20).
- [33] S. Fels, K. Mase, Interactive Video Cubism, in: *Proceedings of ACM International Conference on Information and Knowledge Management*, Kansas City, USA, 1999, pp. 78–82.
- [34] GL4Java Website. http://www.jausoft.com/products/gl4java/gl4java_main.html (last visited: 2003–03–20).
- [35] H. Eidenberger, C. Breiteneder, Visual similarity measurement with the feature contrast model, in: *Proceedings of SPIE Storage and Retrieval for Media Databases*, Santa Clara, USA, 2003 (published on CD).
- [36] H. Frei, S. Meienberg, P. Schauble, The perils of interpreting recall and precision, in: N. Fuhr (Ed.), *Information Retrieval*, Springer, Berlin, 1991, pp. 1–10.
- [37] J.S. Payne, L. Hepplewhite, T.J. Stonham, Evaluating content-based image retrieval techniques using perceptually based metrics, *SPIE Transactions* 3647 (1999) 122–133.
- [38] C.E. Osgood, G.J. Suci, B.H. Tannenbaum, *The Measurement of Meaning*. University of Illinois Press, Urbana, 1971.
- [39] H. Eidenberger, C. Breiteneder, An experimental study on the performance of visual information retrieval similarity models, in: *Proceedings of IEEE Multimedia Signal Processing Workshop*, St. Thomas, US Virgin Islands, 2002 (published on CD).

- [40] OpenML Website. <http://www.khronos.org/> (last visited: 2003-03-20).
- [41] DirectX Website. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wcegmm/htm/dshow.asp> (last visited: 2003-03-20).
- [42] H. Eidenberger, C. Breiteneder, Semantic feature layers in content-based image retrieval: implementation of human world features, in: Proceedings of International Conference on Control, Automation, Robotics and Computer Vision, Singapore, 2002 (published on CD).
- [43] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, SOM-PAK: The Self-organizing Map Program Package, HUT Technical Report, Helsinki, Finland, 1995.
- [44] C. Nastar, M. Mitschke, C. Meilhac, Efficient Query Refinement for Image Retrieval, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Santa Barbara, USA, 1998, pp. 547-552.
- [45] M. Wood, N. Campbell, B. Thomas, Iterative refinement by relevance feedback in content-based digital image retrieval, in: Proceedings of ACM Multimedia Conference, Bristol, UK, 1998, pp. 13-20.