MODELLING OF VISUAL FEATURE DERIVATION IN THE VIZIR FRAMEWORK

Horst Eidenberger

Institute of Software Technology and Interactive Systems, Vienna University of Technology
Favoritenstrasse 9-11, A-1040 Wien, Austria (Europe)
phone: +43 1 58801 18853, fax: +43 1 58801 18898, email: hme@ims.tuwien.ac.at
web: www.ims.tuwien.ac.at

ABSTRACT

If visual information retrieval should make further progress, it will be necessary to identify new ways to derive visual properties from higher levels of understanding than the pixel level (e.g. from low-level features). The paper outlines the implementation of modelling of feature hierarchies in the visual information retrieval framework VizIR (free under GPL). The approach allows for the derivation of high-level features from low-level features by aggregation and localisation as well as semantic enrichment with additional knowledge. The technical implementation is based on the MPEG-7 structures for aggregation and specialisation.

1. INTRODUCTION

Visual information retrieval (VIR) research is driven by the desire to derive semantically meaningful information directly from the content of media objects. Recent years have seen considerable efforts to overcome the enormous gravity of the pixel and to extract more than just colour histograms, edge information and other low-level features. The MPEG-7 standard [1] supports these efforts in two ways: Firstly, it provides aggregation and localisation structures for spatio-temporal feature enrichment. Secondly, it standardises cookbooks for the most prominent low-level features and, by that, provides a foundation for semantic enrichment of low-level features.

Below, we will regard aggregated/localised and semantically enriched as two relevant types of high-level features. The paper describes the way high-level feature structures (e.g. based on MPEG-7 descriptors) can be implemented in the VIR framework VizIR. The VizIR project [2] aims at providing VIR researchers with a workbench of feature extraction, querying and evaluation tools.

The paper is organised as follows: Section 2 gives background information on relevant MPEG-7 models and principles of semantic feature design. Section 3 shortly sketches the VizIR project. Section 4 describes the architecture of high-level feature modelling in VizIR. Additionally, Subsection 4.4 covers important implementation aspects.

2. BACKGROUND

2.1 MPEG-7 aggregation and localisation structures

The visual part of the MPEG-7 standard (e.g. [4]) provides a set of content-based descriptors for colour, texture, shape and motion properties of visual media objects. In addition, it allows for using static (image) descriptors on video content and for spatial as well as temporal localisation of descriptors in media objects.

For temporal aggregation, MPEG-7 provides the *Time Series* descriptor. The *Time Series* descriptor can be used for regular time series (constant interval between samples; e.g. *Colour Layout* descriptions of key-frames) and irregular time series. In the latter case a media object is sub-sampled at the minimum frequency and samples at requested points in time are selected. MPEG-7 describes how *Time Series* have to be extracted and how they have to be compared. Distance measurement is non-trivial for irregular time series.

The main construct for localisation is the *Spatio-Temporal Locator*. It describes object trajectories over space and time. *Spatio-Temporal Locators* exist as *Figure Trajectories* for non-rigid objects and as *Parameter Trajectories* for rigid objects. Both descriptors make use of *Region Locators* and the *Temporal Interpolation* descriptor. *Region Locator* is a simple contour-based description of an object (bounding rectangle, ellipse or polygon). *Temporal Interpolation* is used to estimate unavailable samples by assuming a particular interpolation function. *Figure Trajectories* describe moving objects by the trajectories of object vertices. *Parameter Trajectories* assume motion models for objects and describe object movements by parameter sets.

2.2 Semantic feature enrichment

As human beings we derive our visual similarity perception from generally perceived (e.g. colour distributions) and specifically perceived (recognized) media properties. Today's VIR systems are only capable of extracting generally perceived (low-level) features. Therefore, retrieval results are often unsatisfactory for us. To overcome this shortcoming, considerable research effort is undertaken to enrich low-level features with semantic information.

Generally, three sources of information are available to enhance features: (1) information on the application domain (e.g. on media content), (2) information on the user (e.g. retrieval preferences) and (3) information on the characteristics of the descriptors used (e.g. statistical properties). Technically, additional knowledge can be induced by methods from statistics, artificial intelligence (e.g. neural networks), etc. For example, domain knowledge about football could be used to identify ball and players from shape features (e.g. circularity, edge distribution).

3. THE VIZIR PROJECT

The next three subsections describe the VizIR project in general and, in greater detail, the environment, in which we are applying descriptor aggregation, localisation and enrichment techniques to enhance features and improve content-based retrieval results.

3.1 Overview

The VizIR framework is an open and extendible software workbench for content-based video and image retrieval. It implements state of the art technologies such as the visual MPEG-7 descriptors, the Multimedia Retrieval Markup Language [5] for loose coupling of query engines and user interfaces, etc. and novel paradigms for feature extraction, querying (e.g. 3D user interfaces for integrated browsing and retrieval) and evaluation.

VizIR is free software: it is released as source code under GNU Public License. The VizIR project intends to provide a common software platform that can be used by researchers as a workbench for further VIR research, by software engineers for the development of VIR components for media and database applications and by lecturers for teaching VIR concepts. The VizIR framework was carefully designed to guarantee that these requirements can be met.

Technically, the VizIR framework components are based on the Java programming language, the Java SDK, the Java Media Framework for media processing and other freely available software libraries. The current status of the project, software releases, documentation and development resources can be found on the project website [6]. VizIR is an open project: new users and contributors are always welcome.

3.2 General framework design

The most important part of VizIR is a class framework for feature extraction, querying, refinement, VIR user interfaces, communication, evaluation and benchmarking. Central element is a service kernel. This class is responsible for media administration and query execution. It communicates with query engines, user interfaces and media databases through XML messages (based on the Multimedia Retrieval Markup Language, MRML [5]) and web services.

Media access is hidden in a class that enables accessing the view of visual media (at arbitrary resolution and based on an arbitrary colour model) at any point in time. By that, images and videos can be accessed with a uniform API. Descriptors and media renderer classes make use of the media access class to derive features and to visualise media objects in user interfaces. All elements of user interfaces (including query definition and refinement panels, metadata panels, sketch drawing panels, etc.) are modelled as independent classes that interact with each other through well-defined events and listener methods (locally) or through XML-based web services (remotely). They can be combined arbitrarily to create VIR user interfaces and easily be supplemented by additional querying methods.

VizIR query engines are derived from a common model. This model defines how queries are executed technically (not how the querying logic works). This includes the interface to MRML communication classes, the service kernel and paradigms for database access. All VizIR components are based on an object-oriented database model: If desired, the resources of instances of any class in the framework can be serialised and kept persistent in a database. This feature is guaranteed by an underlying persistence system. The object-oriented database system may be chosen arbitrarily. In our test environment we are currently using a relational database (MySQL) and an object mapping tool (Hibernate). See [2] and [3] for more information on the VizIR architecture.

3.3 Visual feature modelling

Implementation of visual descriptors is a key requirement in the VizIR framework. Descriptor data should be calculated directly from media content and it has to be guaranteed that the descriptor designer does not have to care for storing the descriptor data in the database. VizIR provides two types of classes for the implementation of visual descriptors:

- *DescriptorInfo* classes
- *DescriptorLogic* classes

Derivates of *DescriptorInfo* classes hold the (XML) descriptions extracted by features. These classes have a unique name and their objects have unique IDs. Objects can be stored to and read from a database. *DescriptorInfo* classes are also factories that can be used to create their associated *DescriptorLogic* objects.

DescriptorLogic classes contain just two methods: *extractFeature()* to extract feature data from given media content and *calculateDistance()* to measure distance to a second instance of the same descriptor (a *DescriptorInfo* object). Since *DescriptorLogic* classes contain no relevant status information, they do not need to be made persistent and may have arbitrary names.

The VizIR framework can easily be extended with new descriptors by creating a new *DescriptorInfo* class, creating a new mapping (if the default mapping is not sufficient) and implementing a private *DescriptorLogic* class with the feature extraction logic. New descriptors can be used as soon as the database mapping (an XML document) is available.

4. FEATURE DERIVATION MODELLING

Below, it will be discussed how the described descriptor models can be adapted to allow for feature aggregation, localisation and semantic enrichment. Additionally, in Subsection 4.4 relevant implementation details will be sketched.

4.1 Overview

Generally, descriptors could simply be added to the framework as new classes that make use of already existing algorithms. To enable this scenario, the algorithms used in *DescriptorLogic* classes (e.g. time to frequency transformations, edge operators) would have to be implemented with a well-designed configurable API and collected in software libraries. Still, for the sake of transparency, good software design and maximum code reuse, it would be desirable to have more sophisticated descriptor extension mechanisms available.

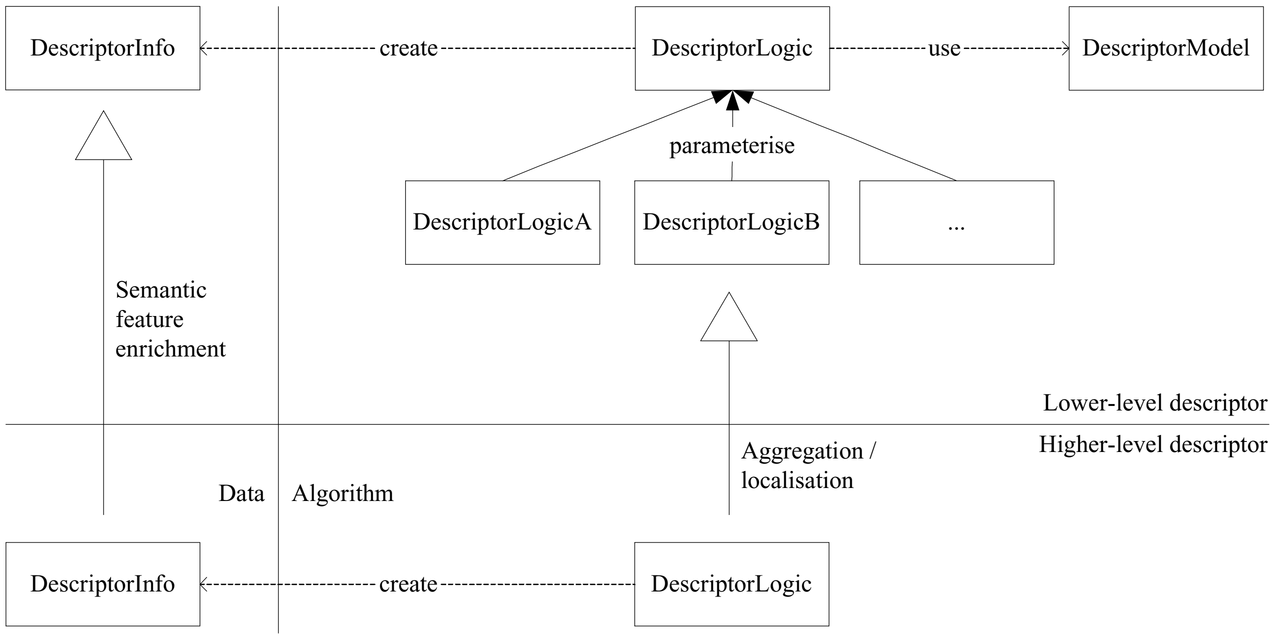

Figure 1: General architecture of high-level feature modelling in VizIR.

Figure 1 describes the general architecture of descriptor design and derivation in the VizIR framework. The left side of the figure shows the data classes. The right side shows the algorithms. The top half shows a low-level descriptor. The bottom half shows a derived descriptor. The design depicted in the figure will be discussed in the next two subsections. Essentially, the idea is that semantically enriched descriptors should be derived from data classes, since they may use completely different algorithms. Aggregated and localised descriptors should be derived from the algorithm classes, because they produce different data classes by using the same or similar algorithms.

4.2 Feature aggregation and localisation

In VizIR, descriptors are stored to databases through an object-oriented mapping technique. Instances of descriptors containing media-specific data are serialised to BLOBs. The BLOBs are referenced by unique IDs (including the class name). For this approach it is necessary that descriptors that create specific output have unique names.

Unfortunately, for example, in the MPEG-7 standard, visual descriptors can be configured to create different descriptions (e.g. *Scalable Colour* histograms may have 32 or 64 bins). If the descriptor classes creating these descriptions would be of the same name and be configured for specific use (e.g. by getters/setters), incompatible descriptions could not be distinguished in VizIR. Therefore, as a first step for aggregation/localisation it is necessary to find an appropriate solution for descriptor configuration.

The solution in VizIR, shown in the top right of Figure 1, is straight-forward: The fundamental description procedure is implemented in a *DescriptorLogic* class. This class may be defined as being abstract. Optionally, the algorithmic details may be laid down in a separate *DescriptorModel* class (e.g. transformations, visual operators). For parameterisation, the *DescriptorLogic* class has a constructor with all relevant parameters. To use a specific version of the descriptor it is sub-classed with a default descriptor that calls the super descriptor with the desired parameters (e.g. histogram size). Since the sub-class needs to have a unique name, its instances can easily be stored to a database and be distinguished from other versions of the same descriptor.

The same pattern is used for aggregation and localisation. This feature is implemented as a parameter of type *Spatio-Temporal Locator* (MPEG-7; in VizIR: a Java class). Then, basically, a spatially and/or temporally specific version of a descriptor is a sub-class of the general descriptor algorithm. In case of aggregation, if an additional envelope is required (e.g. a *Time Series* container or a *Figure Trajectory*, see Subsection 2.1) it is necessary to overload the corresponding *DescriptorInfo* class of *DescriptorLogic* as well. Again, since the class name of the derived class is unique, this means no problem for the persistence system.

Figure 2 shows a further case of simple localisation and aggregation. If the specialised feature is the result of sub-sampling of existing descriptor data, the aggregation/localisation process can be performed by an XSL transformer encapsulated in a *DescriptorLogic* class.

4.3 Semantic feature enrichment

In contrast to aggregation and localisation, completely different workflows and algorithms are used for semantic enrichment of features. In consequence, it would not make sense to derive semantic descriptors on the algorithm level (*DescriptorLogic*). In VizIR, semantic feature classes are descriptions (*DescriptorInfo* objects) derived from existing descriptions. Figure 3 sketches this process. Descriptor data and additional knowledge are used to create semantic features. If the enrichment process is linear and all required

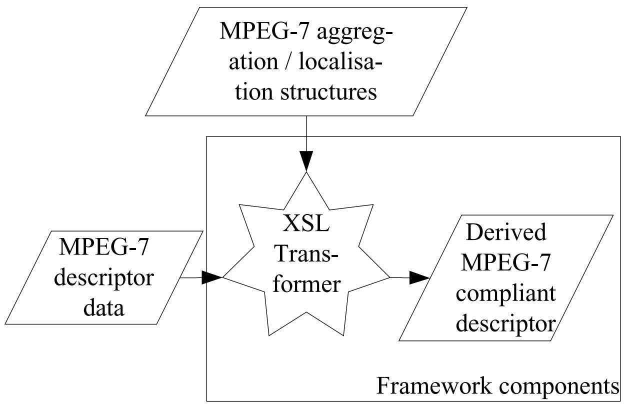
MPEG-7 aggreg-ation / localisa-tion structures
MPEG-7 descriptor data
XSL Trans-former
Derived MPEG-7 compliant descriptor
Framework components

Figure 2: Simple aggregation/localisation of features.

information is available as XML structured data, again, an XSL transformer is sufficient to create the semantic feature.

If a more sophisticated logic (e.g. a statistical procedure) is applied for semantic enrichment, it is recommended to embed the enrichment process in a *DescriptorLogic* class that is not derived from the feature extraction classes used for the low-level features. This paradigm guarantees the existence of reasonable data and logic classes for any type of feature. Consequently, arbitrarily long chains of semantic enrichment, aggregation and localisation of features can be modelled and implemented in VizIR.

4.4 Implementation details

VizIR is based on the Java programming language. Although persistence management is based on an object-oriented database model, it is an important design goal that descriptors are stored into the database in a valid XML format (e.g. the MPEG-7 schemes). Non-VizIR applications (e.g. XSL transformers) have to be able to read the descriptions as well. Internally, descriptions are represented by appropriate memory structures (essentially, one class per tag). Handling is based on the Document Object Model (DOM) of the World Wide Web Consortium. The *JDom* package is used for description parsing and writing. To guarantee that the database requirement is met, *JDom* is used to (redundantly) serialise the description DOMs to string variables. These strings are stored in the database and used as starting point for semantic enrichment of features.

As pointed out above, XSL transformers could solve some of the problems associated with feature aggregation, localisation and enrichment. At the moment, several frameworks for XSL transformation are available and under development. We are currently preferring the Cocoon framework of the Apache project [7], as it is Java-based, in a mature state and easy to install and use. For the applications proposed above, implementing an appropriate transformation style-sheet would be sufficient.

5. CONCLUSION

In this paper, we sketch solutions for the manipulation of visual features in the visual information retrieval framework

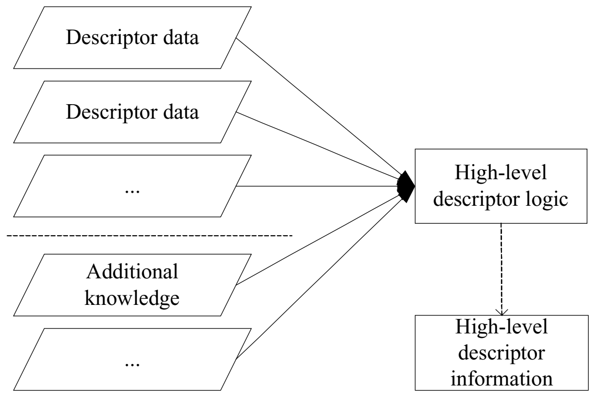
Descriptor data
Descriptor data
...
Additional knowledge
...
High-level descriptor logic
High-level descriptor information

Figure 3: Semantic enrichment of features in VizIR.

VizIR. Considered manipulations are aggregation of feature data over time, localisation in time and/or space and semantic enrichment of features with additional knowledge. Solutions are software patterns for the derivation of feature logic classes and data classes as well as interfaces to helpful external components (e.g. XSL transformers). The presented solutions will be implemented in the VizIR framework and be made available to the VIR community as free software.

ACKNOWLEDGEMENTS

The author would like to thank Christian Breiteneder and Roman Divotkey for their valuable suggestions for improvement. The VizIR project is supported by the Austrian Scientific Research Fund FWF und grant number P16111-N05.

REFERENCES

[1] S.F. Chang, T. Sikora and A. Puri, "Overview of the MPEG-7 standard", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 688-695, June 2001.

[2] H. Eidenberger and C. Breiteneder, "VizIR – A Framework for Visual Information Retrieval", *Journal of Visual Languages and Computing*, vol. 14, pp. 443-469, September 2003.

[3] H. Eidenberger, "Media Handling for Visual Information Retrieval in VizIR", in *Proc. SPIE Visual Communications and Image Processing Conference 2003*, SPIE vol. 5150, July 2003, pp. 1078-1088.

[4] B.S. Manjunath, J.R. Ohm, V.V. Vasudevan and A. Yamada, "Color and texture descriptors", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 703-715, June 2001.

[5] University of Geneva, Multimedia Retrieval Markup Language website, http://www.mrml.net/, last visited 2004-01-07.

[6] Vienna University of Technology, VizIR project website, http://vizir.ims.tuwien.ac.at/, last visited 2004-01-07.

[7] Apache project, Cocoon framework website, http://cocoon.apache.org/, last visited 2004-01-07