MRML: A Communication Protocol for Content-Based Image Retrieval

Wolfgang Müller*, Henning Müller*, Stéphane Marchand-Maillet*, Thierry Pun*, David McG. Squire′, Zoran Pečenović″, Christoph Giess‴, Arjen P. de Vries^4

*Computer Vision Group, Computer Science Department, University of Geneva, Geneva, Switzerland.
′Computer Science and Software Engineering, Monash University, Melbourne, Australia
″LCAV and Ergonomics Group, Ecole Polytechnique Fédérale de Lausanne, Switzerland
‴ Medical and Biological Informatics, Deutsches Krebsforschungszentrum, Heidelberg, Germany
^{4}Department of Computer Science University of Twente Twente, The Netherlands
`Wolfgang.Mueller@cui.unige.ch`
`http://www.mrml.net`

Abstract. In this paper we introduce and describe the Multimedia Retrieval Markup Language (MRML). This XML-based markup language is the basis for an open communication protocol for content-based image retrieval systems (CBIRSs). MRML was initially designed as a means of separating CBIR engines from their user interfaces. It is, however, also extensible as the basis for standardised performance evaluation procedures.
Such a tool is essential for the formulation and implementation of common benchmarks for CBIR. A common protocol can also bring new dynamics to the CBIR field — it makes the development of new systems faster and more efficient, and opens the door of the CBIR research field to other disciplines such as Human-Computer Interaction. The MRML specifications, as well as the first MRML-compliant applications, are freely available and are introduced in this paper.

Keywords: Multimedia retrieval, Communication protocol, Evaluation framework, Reusable software components

1 Introduction

Almost every content-based image retrieval system (CBIRS) is a hard-wired connection between an interface and the functional parts of a program. Some programs provide easy-to-use web interfaces [1], while others need to be installed locally [2] and may be specific to particular operating systems. The reuse of components in CBIR, *e.g.* user interfaces, is thus very sparse. This is not only a time-consuming problem, since everything needs to be developed anew for each system, but it makes the sharing of user data and the comparison of system performances difficult.

In order to address these problems, Y.-C. Chang *et al.* [3] proposed a query taxonomy for multimedia databases. They proposed an initial formulation of the requirements for a system enabling communication between multimedia databases and clients. However, this approach is not yet translated into an extensible protocol.

In this paper we present the Multimedia Retrieval Markup Language (MRML): an XML-based markup language for multimedia queries. MRML was designed to facilitate a bottom-up development approach, which separates the communication problem from the search for the best query language for multimedia databases. In other words, not only it is designed to fulfil the short-term needs of the image database research community, but it is also designed to cater for its long-term needs.

The development of standard query languages, together with standard methods for transmitting queries and data, can improve the interoperability of CBIRSs and thus increase the use and usefulness of multimedia databases. SQL and ODBC are examples of such developments for relational databases. The aim of MRML, however, is more similar to that of the DICOM protocol [4], which promoted the interoperability of medical imaging systems from different vendors. In summary, we address the urgent need for common tools which will facilitate the development and evaluation of multimedia database systems. By this means, we aim to facilitate the development of common benchmarks for CBIRS performance, similar those used for textual information retrieval [5].

The query-by-example (QBE) paradigm with relevance feedback (including browsing) is the search paradigm employed by most current CBIRSs. We therefore provide an extensible QBE facility within MRML. Further, some MRML-compliant tools have been developed and made freely available under the GNU Public License (`http://www.mrml.net/download/`). These are described briefly in Section 2, and include a CBIR search engine (*Viper*), which acts as a server, and an interface (SnakeCharmer), which acts as a client. Scripts (mostly Perl scripts) have also been made available, which might provide a basis for the creation of standard CBIRS benchmarks. An overview of various evaluation methods is given in [6], where the use of freely-available annotated image collections, such as [7], as test datasets is also advocated.

In order to be useful for research, MRML needs to be a “living standard”: research groups will need to be able to test and use extensions without having to ask a committee for approval. We therefore employ a development model which permits phases of independent growth with subsequent code merging. In § 3, we present the main features of MRML and, in § 4, we show an example of how MRML can be extended to suit particular needs while staying coherent with the common standard.

2 *Viper*, CIRCUS and SnakeCharmer

MRML was initially designed to facilitate cooperation between research groups. The main programs for our testbed originate from the Ecole Fédérale Polytechnique de Lausanne (CIRCUS and SnakeCharmer) and from the University of Geneva (*Viper*). In this testbed, we use MRML to link a single interface (SnakeCharmer) to two different CBIRS (CIRCUS and *Viper*).

Viper [1] is an image search engine based on techniques commonly used in text retrieval and thus offers efficient access to a very large number of *possible* features (more than 80,000 simple colour and texture features, both local and global). Each image contains only a subset of these features. Access to images containing given features is provided by an inverted file, a standard access technique in text retrieval. The emphasis in *Viper* is on adapting the system response according to interaction with a user – positive and negative relevance feedback is accepted over several steps. Detailed descriptions of *Viper* may be found in [8, 9].

CIRCUS [2] is a server framework supporting multiple image retrieval methods and algorithms.

Currently four methods are implemented. The first applies an adaptation of Latent Semantic Indexing [10] to image features describing local and global colour and texture, as well as global layout and optional keywords. The second is a texture/layout-specific method based on wavelet maxima moments. It extracts a set of contours from the image at various levels of detail, invariant to scale, translation, and partially to illumination changes. The third approach is texture-specific, it describes textures by computing the parameters of a Hidden Markov Model governing the coefficients of a wavelet decomposition of a textured image. The similarity is evaluated using the Kulbach-Leibler distance between two distributions. The last method is a fast, wavelet packet-based, approximation of the Principal Component Analysis, based on the features used by the other methods. It is the most scalable and fastest of the implemented methods.

SnakeCharmer (figure 1) is an MRML-compliant client application. It is written in JAVA for portability and offers query by multiple positive and negative examples, query history, multiple collection and algorithm selection, a scatter plot of the results according to various aspects of similarity and a basket for user-selected images.

3 Multimedia Retrieval Markup Language

MRML[3] is formally specified in [11]. It provides a framework that separates the query formulation from the actual query shipping. It is designed to markup multi-paradigm queries for multimedia databases. MRML enables the separation of interface and query engine and thus eases their independent development.

[1] http://viper.unige.ch/

[2] http://lcavwww.epfl.ch/CIRCUS

[3] http://www.mrml.net

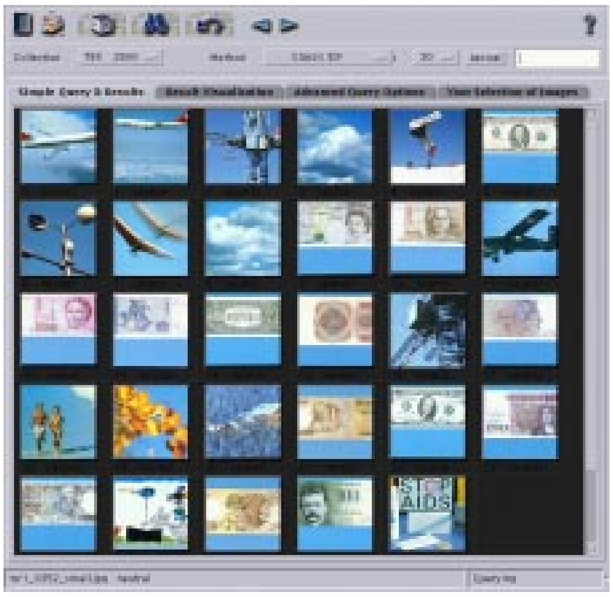


Fig. 1. The JAVA interface SnakeCharmer

MRML can be embedded into an existing system with little effort. First, it is XML-based, meaning that standard parsers can be used to process the communication messages. Further, the code for an example MRML-compliant CBIR system is freely-available and provides the basic implementation of both ends of an MRML-based communication toolkit. MRML is currently in a testing phase at several universities and further applications based on this protocol such as benchmark systems and meta-query engines are under development.

MRML is designed to allow extension by independent groups. By this means, it provides a research platform for extensions which later may become a part of common MRML.

3.1 Design goals of MRML

It is important for the following sections to keep in mind the priorities which we took into account during the design of MRML.

Interoperability: Interoperability is an obvious short term need of the CBIRS community. The fact that the interface between CBIRS client and server is not specified hampers research. Topics that could benefit from interoperability include:

- Meta-query engines query several "normal" query engines and assemble the results [12]. Constructing a meta-query engine would require to define a protocol abstraction layer corresponding to each of the different query embedded in the system. Using a common protocol would save a substantial amount of work.

  - Human-computer-interaction aims at comparing the impact of different user interfaces on the performance of identical query engines, or test several engines with the identical interfaces. In this context, by ensuring the compatibility between engines and interfaces, MRML would ease this type of evaluation.
  - Evaluation of query engines: Thanks to MRML, one can design a benchmark package that connects to a server, sends a set of queries and evaluates the results.

Extensibility without administration overhead: it was our goal to provide a communication protocol which can be extended without having to ask a standardisation body for permission. MRML enables independent development of extensions. As we will describe in Section 4.1 we invite MRML users to render their extensions accessible at `http://www.mrml.net/extensions/`. Later, stable extensions can be added to new common versions of MRML.

Common log file format: The whole area of CBIRS is craving for ground truth or other user data. MRML provides a common, human readable, easy to analyse, format for logging communication between CBIRS client and server. MRML contains a maximum of data which might be of interest for computer learning purposes. If needed, extensions of MRML can be designed in order to send additional data.

Simplicity of implementation: everything was designed so as to minimise the implementation overhead incurred when using MRML, while keeping a maximum of flexibility. MRML only uses a subset of the features of XML in order to maximise the number of tools that can use MRML.

3.2 Features of MRML

MRML-based communications have the structure of a remote procedure call: the client connects to the server, sends a request, and stays connected to the server until the server breaks the connection. The server shuts down the connection after sending the MRML message which answers the request. This connectionless protocol has the advantage of easing the implementation of the server. To limit the performance loss caused by frequently reconnecting, it is possible to send several requests as part of a single MRML message. The extension of MRML to a protocol permitting the negotiation of a permanent connection is also planned.

MRML, in its current specification (and implementation) state, supports the following features:

- request of a capability description from the server,
- selection of a data collection classified by query paradigm; it is possible to request collections which can be queried in a certain manner,
- selection and configuration of a query processor, also classified by query paradigm; MRML also permits the configuration of meta-queries during run time,
- formulation of QBE queries,
- transmission of user interaction data.

The final feature reflects our strong belief that affective computing [13] will soon play a role in the field of content-based multimedia retrieval. MRML already supports this by allowing the logging of some user interaction data. In particular, this is the case for the history-forward and history-backward functionalities of the SnakeCharmer interface.

Why XML and not CORBA? There are important reasons for using XML rather than a communication framework such as CORBA as a basis for the implementation of MRML. The first is that when using XML no large communication framework is necessary, as it is for CORBA. Secondly, MRML offers a common human-readable format for log files. Programming and debugging issues aside, having a simple common format for user data will make it easier for research groups to share this type of data. Together with common free image collections, MRML-compliant systems will form a powerful tool for collecting and sharing CBIR user interaction data.

Another reason for the use of XML as a basis for MRML is the large number of free XML tools available such as parsers and tools to evaluate files in XML format (XML Query Language). XML is about to become the main description language for all kinds of meta data on the Internet and may also be used in MPEG-7 [14], thus ensuring the long-term support of its specifications.

Graceful degradation: independent development on a common base Graceful degradation is the key to successful independent extension of MRML. The basic principles can be summarised as follows:

- servers and clients which do not recognise an XML element or attribute encountered in an MRML text should completely ignore its contents,
- extensions should be designed so that all the standard information remains available to the generic MRML user (see examples in Section 4).

These principles provide guidelines for independent extensions of MRML.

To avoid conflicts between differing extensions of MRML, and in order to we plan to maintain or promote a central database for the registration and documentation of MRML extensions. This would also facilitate the "translation" between user logs which contain extended MRML.

3.3 Logging onto a CBIR server

An MRML server listens on a port for MRML messages on a given TCP socket. When connecting, the client requests the basic properties of the server, and waits for an answer. Skipping standard XML headers, the MRML code looks like this:

```
<mrml>
 <get-server-properties />
</mrml>
```

The server then informs the client of its capabilities. This message is empty in the current version of MRML, but it allows for the extension of the protocol:

```
<mrml>
 <server-properties />
</mrml>
```

Goal of this tag is to provide a stub for negotiation which influences the whole communication, like e.g. the opening of a permanent, possibly encrypted connection.

Further negotiation between client and server may depend on the user, so before further negotiation we have to open a session for the user:

```
<mrml>
 <open-session
  user-name="A. User" session-name="a session" />
</mrml>
```

which will be answered by an acknowledgement signal containing the ID of the session just opened. We regard the concept of sessions as very important, as it allows multi-user servers and across-session learning. Of course, it is possible to close and rename sessions.

Now one can request a list of collections, which are available on the server user:

```
<mrml session-id="s-33">
 <get-collections />
</mrml>
```

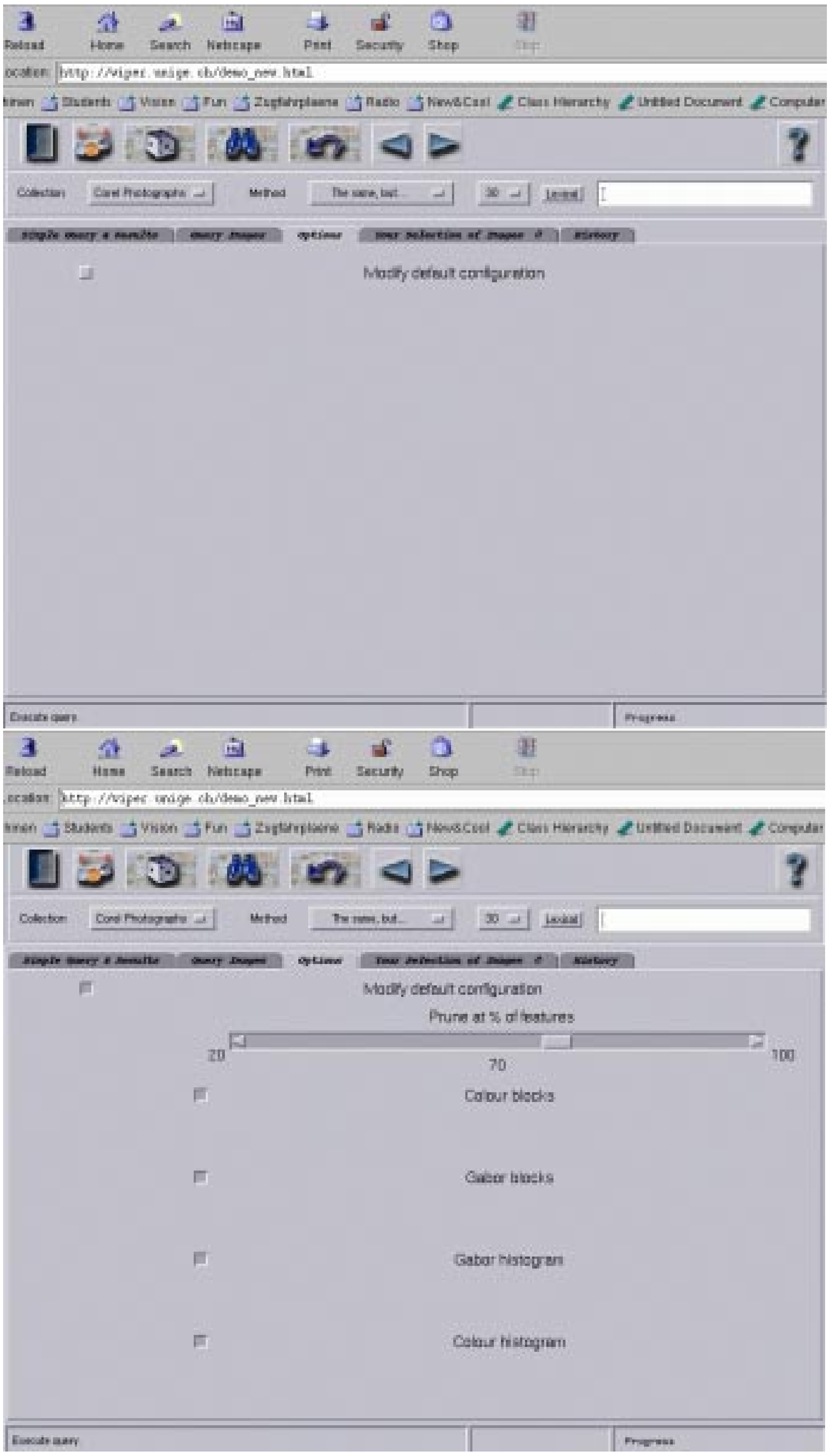
The answer will be a list of collections, with a description of the ways the collection has been indexed, encapsulated in a `query-paradigm-list` tag.

Similarly, the client can request a list of algorithms (*i.e.* query methods), which can be used on the server. Each of the `algorithm` tags returned also contains a `query-paradigm-list` describing the way the algorithm can interact with the user, as well as which indexing methods are needed for employing the algorithm.

The user is now able to choose on his client an algorithm/collection combination which suits his needs, and in which the `query-paradigm-list`s of collection and algorithm have *match*. The matching of `query-paradigm-list`s is described in [11].

3.4 Interface configuration

The client can then request property sheet descriptions from the server. Different algorithms will have different relevant parameters which should be user-configurable (*e.g.* feature sets, speed vs. quality). *Viper*, for example, offers several weighting functions [15] and a variety of methods for, and levels of, pruning. All these parameters are irrelevant for CIRCUS. Thanks to MRML property sheets, the interface can adapt itself to these specific parameters. At the same time, MRML specifies the way the interface will turn these data into XML to send them back to the server. The interested reader is referred to [11] for details.



Location: http://viper.unige.ch/demo_new.html

Collection: Corel Photographs

Method: The same, but...

30

Modify default configuration

Prune at % of features: 20 – 70 – 100

Colour blocks

Gabor blocks

Gabor histogram

Colour histogram

Fig. 2. Demonstration of property sheets in SnakeCharmer. The user has the choice to modify the default settings or not. If he decides to modify the default settings, widgets which enable him to do so pop up.

3.5 Query Formulation

The query step is dependent on the query paradigms offered by the interface and the search engine. MRML currently includes only QBE, but it has been designed to be extensible to other paradigms.

A basic QBE query consists of a list of images and the corresponding relevance levels assigned to them by the user. In the following example, the user has marked two images, the image `1.jpg` positive (`user-relevance="1"`) and the image `2.jpg` negative (`user-relevance="-1"`). All query images are referred to by their URLs.

```
<mrml session-id="1" transaction-id="44">
<query-step session-id="1"
 resultsize="30"
 algorithm-id="algorithm-default">
 <user-relevance-list>
  <user-relevance-element
   image-location="http://viper.unige.ch/1.jpg"
   user-relevance="1"/>
  <user-relevance-element
   image-location="http://viper.unige.ch/2.jpg"
   user-relevance="-1"/>
 </user-relevance-list>
</query-step>
</mrml>
```

The server will then return the retrieval result as a list of images, again represented by their URLs.

Queries can be grouped into transactions. This allows the formulation and logging of complex queries. This may be applied in systems which process a single query using a variety algorithms, such as the split-screen version of *TrackingViper* [16] or the system described by Lee *et al.* [17]. It is important in these cases to preserve in the logs the knowledge that two queries are logically related one to another.

4 Extending MRML

How to extend In order to demonstrate how easily MRML can be extended to other query paradigms, we give as an example QBE for images with user annotation. We assume that the user is invited to associate textual comments with images he or she marks as relevant or irrelevant. Since a tag for this purpose does not yet exist in MRML, we add an attribute `cui-user-annotation="..."` to the element. The prefix `cui-` is added to avoid name clashes with extensions from other groups which use MRML (namespaces are avoided here to keep things simple for old XML parsers).

```
 <user-relevance-list>
  <user-relevance-element
   image-location="file:/images/1.jpg"
  user-relevance="1"
  cui-user-annotation="tropical fish"/>
</user-relevance-list>
```

It is important to note here that servers which do not recognise the attribute `cui-user-annotation` still can make use of the remaining information contained in the `user-relevance-element` element.

How *not* to extend As an example of how *not* to extend MRML, we give an extension with the same semantics but which does not respect the principle of graceful degradation:

```
<user-relevance-list>
 <cui-user-relevance-element
  image-location="file:/images/1.jpg"
  user-relevance="1"
  user-annotation="tropical fish">
</user-relevance-list>
```

Instead of adding an *attribute* to an existing MRML element (in this case, `user-relevance-element`), a new *element* was defined that contained the same kind of extension, namely `cui-user-relevance-element`. Consequently, servers which do not recognise this element will not be able to exploit any relevance information.

MRML and Binary data MRML's preferred mechanism for transferring binary data is to send the URL where the data can be found. Binary data is then retrieved using the URL. As it is a primary goal of MRML to enable the sharing of logging data we suggest to transfer big chunks of data as follows.

Binary data which stays constant over several sessions (*i.e.* images and other media items contained in the queried collection) should be transfered using their URL, as described above. This keeps log files relatively small, yet data is accessible for everyone.

Binary data which changes during the query process (*e.g.* a file containing an example image for a QBE query which is not accessible by the web) should be transferred using two attributes. One of the attributes should contain the base64-encoded binary data, the other one the corresponding MIME type.

However, in most cases, it is preferable to design proper extensions to MRML which provide the best accessibility and readability of the resulting logs.

4.1 The MRML development model

As it has been stated many times throughout this article, MRML allows each developer to extend MRML to his needs. In particular, these extensions can coexist, and a notification of a central body is not necessary for making these extensions work. However, to maximise the usefulness of MRML the authors are presently setting up a database which contains documentation of extensions to MRML. It is also intended to provide a forum for groups which want to extend MRML into similar directions.

We propose to develop extensions to MRML in the following fashion. Search first the page `http://www.mrml.net/extensions/` for documentation of extensions which might already do what you want.

- If so,
  1. implement the existing extension
  2. double-check with the author of the existing extension that the documentation has been understood in the right way
  3. add your name and your affiliation to the list of people/groups who are using this extension which is kept on `www.mrml.net`.
- If not,
  1. implement the extension
  2. submit documentation for your extension along with your name and your affiliation to `www.mrml.net`.

The information contained on `http://www.mrml.net/extensions/` will be useful both for analysing logs and for *merging* extensions, once an extension has proven more useful than others.

5 Further use of MRML

In this article, we have presented a stable, extensible and useful framework for the use in CBIRS and other multimedia retrieval systems. In the sequel, we shortly described tools that can be easily implemented using existing features of the MRML framework.

5.1 Meta query engines

We are currently conceiving a meta query engine which queries MRML compliant servers.

Meta query engines running under MRML will start a handshaking procedure, establishing for each of the attaches servers the available collections and algorithms. The meta query can then assemble this information into a property sheet that can be presented to the user via a standard MRML interface.

After configuration the meta query engine will pass arriving queries onto the attached servers, returning an assembled result. We plan to use methods similar to the ones described in [12].

5.2 Benchmarks

Only preliminary steps have been taken by the CBIR community towards developing common benchmarks – a comparison of evaluation techniques may be found in [6]. We are currently working on a more profound and flexible benchmarking system extending the results of this research. See figure 3 for a description of the structure of such a benchmark.

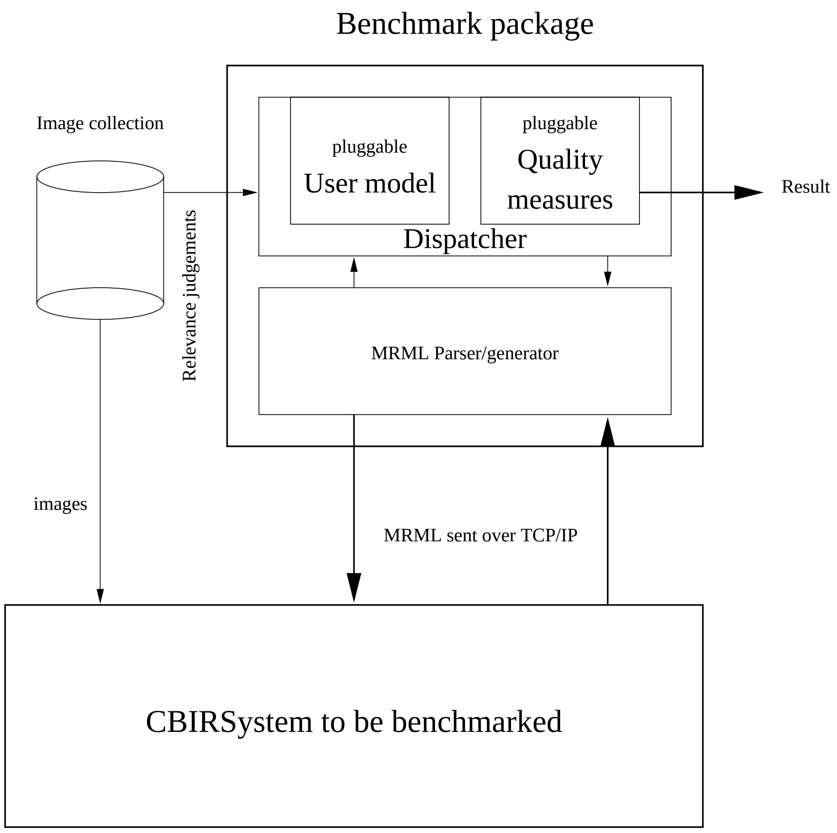


Fig. 3. We propose a benchmark that relies on stored relevance judgements as a basis for the simulation of user feedback. We propose storing the relevance judgements done by several users for a set of identical queries in order to account for the fact that different users will judge relevance differently. The data in [18] have been obtained using this technique.

6 Conclusion

The development of MRML and the first MRML-compliant tools has established a common framework for the fast development of CBIR applications. To our knowledge, MRML is the first general communication protocol for CBIR actually implemented. The source code for the interface and the query engine is freely available. This should help developers of retrieval engines and developers of user interfaces to develop complete systems on the basis of existing components. Extensive tests have shown the stability of the protocol and our test components.

Since MRML is a free and extensible standard, the availability of more applications and tools supporting such a protocol will further facilitate the development of CBIR applications supporting a diversity of query paradigms.

More important, in our opinion, is the fact that the adoption of MRML will lead to the possibility of comparing different CBIR applications objectively. It will make it easy to develop common benchmarks for all MRML-compliant systems, similar to those which exist in the database and information retrieval communities.

Finally, the possibility of sharing MRML user logs will provide a useful tool for the sharing of user interaction data.

Acknowledgements

This project is supported by the Swiss National Foundation for Scientific Research under grant number 2000-052426.97. 10

References

1. Surfimage webdemo. web page: `http://www-rocq.inria.fr/cgi-bin/imedia/surfimage.cgi` (1999)
2. QBIC^{TM} – IBM's Query By Image Content. web page: `http://wwwqbic.almaden.ibm.com/~qbic/` (1998)
3. Chang, Y.-C., Bergmann, L., Smith, J. R., Li, C.-S.: Query taxonomy of multimedia databases. In: Panchanathan *et al.* [19]. (SPIE Symposium on Voice, Video and Data Communications)
4. Revet, B.: DICOM Cook Book for Implementations in Madalities. Philips Medical Systems, Eindhoven, Netherlands (1997)
5. Vorhees, E. M., Harmann, D.: Overview of the seventh text retrieval conference (TREC-7). In: *The Seventh Text Retrieval Conference*. Gaithersburg, MD, USA (November 1998)
6. Müller, H., Müller, W., Squire, D. M., Pun, T.: Performance evaluation in content-based image retrieval: Overview and proposals. Tech. Rep. 99.05, Computer Vision Group, Computing Centre, University of Geneva, rue Général Dufour, 24, CH-1211 Genève, Switzerland (dec 1999)
7. Annotated groundtruth database. Department of Computer Science and Engineering, University of Washington, web page: `http://www.cs.washington.edu/ research/imagedatabase/groundtruth/` (1999)
8. Müller, H., Squire, D. M., Müller, W., Pun, T.: Efficient access methods for content-based image retrieval with inverted files. In: Panchanathan *et al.* [19]. (SPIE Symposium on Voice, Video and Data Communications)
9. Squire, D. M., Müller, W., Müller, H., Raki, J.: Content-based query of image databases, inspirations from text retrieval: inverted files, frequency-based weights and relevance feedback. In: *The 11th Scandinavian Conference on Image Analysis (SCIA'99)*. Kangerlussuaq, Greenland (June 7–11 1999)
10. Pečenović, Z.: Image retrieval using Latent Semantic indexing. Final year graduate thesis, AudioVisual Communications Lab, Ecole Polytechnique Fédérale de Lausanne, Switzerland (June 1997)

11. Müller, W., Pečenović, Z., de Vries, A. P., Squire, D. M., Müller, H., Pun, T.: MRML: Towards an extensible standard for multimedia querying and benchmarking – Draft proposal. Tech. Rep. 99.04, Computer Vision Group, Computing Centre, University of Geneva, rue Général Dufour, 24, CH-1211 Genève, Switzerland (October 1999)
12. Beigi, M., Benitez, A. B., Chang, S.-F.: Metaseek: A content-based meta-search engine for images. In: *Symposium on Electronic Imaging: Multimedia Processing and Applications - Storage and Retrieval for Image and Video Databases VI, IST/SPIE'98, San Jose, CA* (1998)
13. Picard, R. W.: Affective Computing. MIT Press, Cambridge (1997)
14. MPEG Requirements Group: MPEG-7: Context and objectives (version 10 Atlantic City). Doc. ISO/IEC JTC1/SC29/WG11, International Organisation for Standardisation (October 1998)
15. Salton, G., Buckley, C.: Term weighting approaches in automatic text retrieval. Tech. Rep. 87-881, Department of Computer Science, Cornell University, Ithaca, New York 14853-7501 (November 1987)
16. Müller, W., Squire, D. M., Müller, H., Pun, T.: Hunting moving targets: an extension to Bayesian methods in multimedia databases. In: Panchanathan *et al.* [19]. (SPIE Symposium on Voice, Video and Data Communications)
17. Lee, C. S., Ma, W.-Y., Zhang, H.: Information Embedding Based on User's Relevance Feedback for Image Retrieval. In: Panchanathan *et al.* [19]. (SPIE Symposium on Voice, Video and Data Communications)
18. Squire, D. M., Müller, W., Müller, H.: Relevance feedback and term weighting schemes for content-based image retrieval. In: Huijsmans, D. P., Smeulders, A. W. M., eds., *Third International Conference On Visual Information Systems (VISUAL'99)*, no. 1614 in Lecture Notes in Computer Science. Springer-Verlag, Amsterdam, The Netherlands (June 2–4 1999)
19. Panchanathan, S., Chang, S.-F., Kuo, C.-C. J., eds.: Multimedia Storage and Archiving Systems IV (VV02), vol. 3846 of *SPIE Proceedings*, Boston, Massachusetts, USA (September 20–22 1999). (SPIE Symposium on Voice, Video and Data Communications)