A Flexible Image Search Engine[†]

Panrit Tosukhowong[‡]
Frederic Andres
Kinji Ono
NACSIS R&D
Otsuka 3-29-1 Bunkyo-ku
Tokyo 112
81-3-3942-5940
{ono,andres}@rd.nacsis.ac.jp
panrit@net.is.uec.ac.jp

José Martinez
Noureddine Mouaddib
Nicolas Dessaigne[§]
IRESTE/IRIN
40 Christian Pauc La Chantrerie
BP 60601 44306 Nantes
81-3-2-40-68-32-56
{jmartine,nmouaddi}@ireste.fr
ndessaig@ireste.fr

Douglas C. Schmidt
Washington University, CDOC
Bryan Hall, Room 503
Campus Box 1045
One Brookings Drive
St. Louis, Missouri 63130-4899
1-314-935-4215
schmidt@cs.wustl.edu

ABSTRACT

Multimedia searching over Internet has gained substantial popularity in the past two years. Java's networking features, along with the growing number of Web browsers that can execute Java applets, facilitate distributed processing. Networking and computational performances are key concerns when considering the use of Java to develop performance-sensitive distributed multimedia search engines. This paper describes MISE, the MediaSys Image Search Engine over a large-scale network. After an overview of the architecture, we present the search capabilities of MISE as companion part of image processing.

Keywords

MediaSys, image search engine, retrieval by content, fuzzy logic.

1. INTRODUCTION

Lot of works in the field of multimedia management systems have followed generic approaches and have developed visualisation tools in the field of search by content, mainly for images [18] [19] [15] [5]. However, problems of large scale distributed multimedia information systems such as the quality of the search, the performance, the flexibility, and the customisability have been mostly ignored. MediaSys provides such an extensible infrastructure for multimedia management. For instance, plug-ins for image operations or filters can be associated and integrated, even dynamically. This eases its upgrades according to the user's requirements and the evolving technology.

The remainder of this paper is organised as follows: Section 2 overviews the MediaSys platform. Then, Section 3 describes the key features of the MISE client applet. Next, Section 4 details the search process. Finally, we give some concluding remarks.

2. THE MEDIASYS PLATFORM

The large-scale distributed image-based MediaSys system is composed of a set of MediaSys servers and a set of MISE clients. MediaSys servers are multimedia storage systems. Image data produced by various devices are transferred to them. Using MISE client tools, users with various roles and profiles can search for and access to specified image data. Moreever. they can visualise them and/or analyse them.

The MediaSys server is based on the AHYDS platform (*Active HYpermedia Delivery System*) [2], which is an active hypermedia delivery system developed at NACSIS since 1995. MISE is an evolution of Find^{Im}AGE prototype [11] developed at IRESTE/IRIN since 1996. Furthermore, the MISE tool uses the resource manager JACE [20] developed at Washington University, Saint Louis, Missouri. Finally, the design follows the Active Object approach of [8].

3. THE KEY FEATURES OF MISE

MISE allows users to search for and access to any image stored in a MediaSys server. In addition, the MISE applet provides a hierarchical browser that allows users to traverse sets of images on remote MediaSys servers. This makes it straightforward to find and select images across the network, making MISE quite usable, as well as easy to learn. Once an image has been downloaded, image filters or image operations can process it. Although MISE is targeted for distributed image systems, it is a general-purpose image-manipulation engine. Image filters or operations can be dynamically configured into MISE via the Service Configurator pattern [17] without neither reloading nor restarting the MISE applet. Conversely, a processed image can

† This paper has been supported in part by the Japanese Ministry of Education under the COE Framework.

‡ P.h.D. Student of The University of Electro-Communication Tokyo, Japan

§ Visiting student researcher at NACSIS (JAPAN) dessaign@mcsun4.rd.nacsis.ac.jp be uploaded to the image server from where the applet was downloaded thanks to the MediaSys server support.

There are six main components in a MISE client applet:

Graphical User Interface: which provides a front-end to the image search engine and to the image processing tool. It enables the users to search images, to receive images, to process them and to send them back to image servers (MediaSys server.) Figure 1 illustrates the MISE graphical user interface (GUI.)

Server Locator: which locates a URL address associated to the user role that can reference an image or an image server (MediaSys-typed server). If the URL points to a MediaSys server, its content is listed so those users can browse them to choose one image to process.

Image Downloader: which downloads an image or a set of images located by the Server Locator as input of the search process (see in Section 4).

Image Uploader: which uploads the currently displayed image to the server from where the applet was downloaded (applet security restrictions.) In addition, the MediaSys server supports data uploading using the HTTP and its PUT method.

Image Processing Tool: which extract characteristics from the displayed image using either filters, or other image analysis techniques. The end of the image processing is characterized by the display of the resulting image. Figure 2 is a snapshot of this Tool.

Plug-ins Configurator [3]: which downloads dynamically image processing plug-ins such as image filters, edge detectors, colour detectors from the MediaSys Server and configures them in the applet according to the needs of the user. The Filter Configurator uses the Service Configurator pattern [17] to dynamically configure the image filters and image operation.

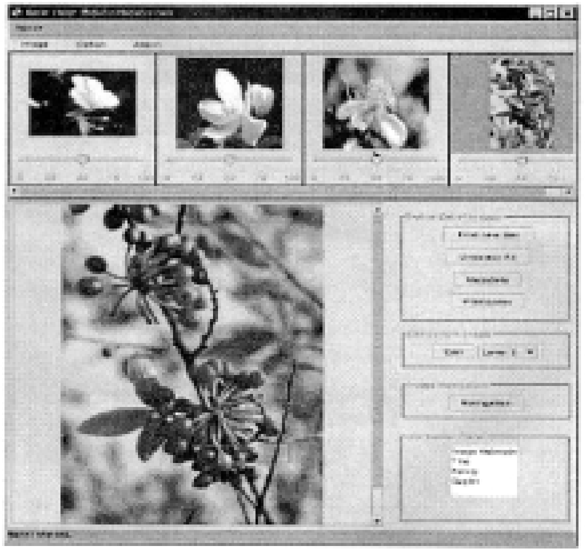

Figure 1. The MISE User Interface

4. THE MISE SEARCH PROCESS

The search process is located at the server side. Its input consists of a set of sample images along with their associated weights as given by the user in the client applet. The purpose of the search engine is to retrieve an ordered list of images of decreasing similarity with respect to the user's choices. Classically, sample images with the highest weight, say 1.0, will always be retrieved (if they are in the database), whereas the ones with the lowest weight, accordingly 0.0, will never appear in the result list. A threshold has to be given in order not to retrieve about the whole database.

Figure 2. The Image Processing Tool

4.1 The Relevance Feedback Loop Approach

From the user's point of view, the process has been extremely simplified. It is based on the information retrieval paradigm [6]. The end-user selects whether the currently displayed images fit his or her need or not. In Figure 1, under each image, a slider allows him or her to give a rank between 0 and 1 (default is 0.5.) The use of example and counter-example images offers advantages. First, this is easier for the end-user because it is more intuitive. Secondly, the graphical interaction is limited to weight annotations of *images*, rather than tuning several unintuitive knobs, as pointed out by other authors [1] [16]. From this input, the system infers automatically one formal query that best fits the user's hints, with respect to the set of properties that have been extracted on each image and activated in the interface. This query is issued onto the database and a new list of images is displayed for another relevance feedback loop. The process continues until the user is satisfied by the returned answer, or that he or she feels that good images are not present in the database [12]. The current version of the system is based on fuzzy logic, though the underlying query language remains OQL (*Object Query Language*) [4].

4.2 The Indexing Process

Each image in the database has to be indexed. The indexing process consists in extracting as much features as possible. Possible features are regions, global histograms for colours or intensity, edges, and even meta-data such as keywords (which can be provided "automatically" through advanced image formats,

e. g., the forthcoming MPEG-7.) Currently, only colour segmentation of regions has been tested for retrieval by contents. Histograms and above all keywords are being added and should provide much more precise searches thanks to semantics [15]. The segmentation process in regions is done with respect to a given colour partitioning. The algorithm and the obtained results have been detailed in [9, 10]. For the sake of retrieval, regions are characterised by a set of unary and binary functions, such as the horizontal position or the adjacency of two regions respectively.

4.3 The Query Inference

The other technical part of the querying process is to find the "best" query to issue on the database related to the given set of sample images. This problem is NP-complete, as demonstrated in [10]. Therefore, an approximate algorithm is required, in order not to incur too high a search time. Our choice has been to use genetic algorithms [7] for several reasons. They offer a good compromise between speed, "exhaustive" search, ease of paralleling in the future among the variety of learning machine and optimisation algorithms [14] [13]. Furthermore, several of the alternative algorithms present some severe drawback with respect to the precise requirements of searching for an OQL query (e. g. "branch-and-bound" was not applicable at all due to the presence of both examples and counter-examples).

In our case, each chromosome represents a possible query. A query is a conjunction of the various functions applied to several regions that are quantified existentially. For each function, we provide an inclusion test, i. e. a disjunction of possible outcomes [11]. Each candidate query (chromosome) is applied to each sample image in order to compute its fitness, i. e., how close to the user's hint it is. The algorithm stops once the best chromosome fitness of a generation has reached a threshold value, or if a maximum number of generations have been computed.

4.4 Fuzzy Queries

Though the introduction of fuzzy logic in the search algorithm is quite time consuming (about 10 times slower than its crisp counterpart), it has some fundamental implications.

Let us consider the horizontal function as an example. In the crisp version, the x centroids of the regions were categorised into only five possible position classes: west, middle west, middle, middle east, and east. In the fuzzy version, we kept the same classification but provided smooth transitions between the classes, i. e. the linguistic variables.

For the sake of illustration, let us use the example described in Figure 3. The shown region is located in the middle east class. However, if a query asks for a region in the middle position, the horizontal position function will not return a null value, but a value close to 0.5.

The used norm and co-norm are the standard max and min operators. Each chromosome is applied on each sample image to obtain its fuzzy membership degree with respect to the represented query.

The best chromosome is the one with the shortest distance between the weights given by the user and the corresponding list of membership degrees. We adopted a mere Euclidean distance, which penalises large differences.

Additionally, all users' hints cannot be considered with the same importance. Intuitively, at the one end, if a user indicates that an image is absolutely comparable (resp. unrelated) to his or her expectations, then this hint must be strongly taken into consideration. At the other end, a weight of 0.5 does not give any information at all about the kind of images that the user is looking for (the default value, as shown in Figure 1.) We use a sinusoidal formula to adjust the quality of an hint. Actually, it is used to adjust the value of a difference, the formula being *difference* = (*weight* - *degree*) $\times$ *quality* (*weight*), which are in turn used in the Euclidean distance computation.

4.5 Crisp logic vs. fuzzy logic

Though the fuzzy implementation is time consuming, it allows more flexibility and solves some problems. First, the user is no longer constrained to a frustrating ternary choice: positive or negative sample, or irrelevant image. The use of weights brings some flexibility, though we had to control its impact through the quality function. Then, the system is much less sensitive to bad positive examples. Effectively, a positive sample must respect *all* the predicates of a query, whereas a negative sample can be satisfied as long as at least *one* predicate fails. Therefore, if the user provides a positive example that is incompatible with the other examples, then the system cannot infer a meaningful crisp query. With fuzzy logic, and the Euclidean distance metrics, some mistakes can be tolerated. Also, thanks to the membership degree, images can be ranked, and the size of the retrieved set can be adjusted with a minimal threshold value.

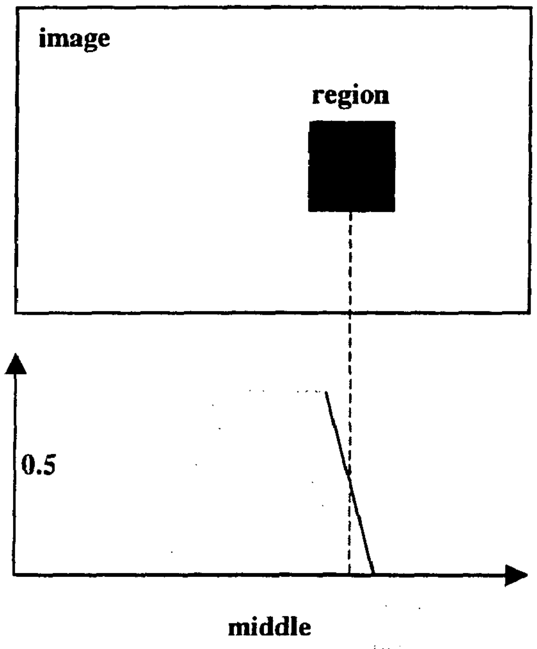

Figure 3. A region, the horizontal position values of which are 1.0 for middle-east, 0.5 for middle, and 0.0 for others

5. CONCLUSION

This paper described the design and the implementation of the MediaSys Image Search Engine (MISE), a multimedia distributed system over Internet and/or high-speed networks. This system enables the users to search, to browse, to process, and to store images according to the combination of visual and textual features with meta-data related to the images. The MediaSys servers store the meta-data, visual and textual features, and the images themselves over a large scale distributed and heterogeneous system.

MISE, the first client tool of the MediaSys project, gains its efficiency by the use of Java. The Java approach enables to build valuable tool as MISE is a simple, portable, and distributed system.

As on-going research work, the MediaSys Image Search Engine is being extended in terms of flexible search algorithms with new image features such as metadata support, fuzzy thesaurus, and multi-resolution. Performance evaluation will assess our approach and the architecture of MISE search algorithm.

6. ACKNOWLEDGEMENTS

We would like to thank NACSIS, IRESTE/IRIN and Washington University/CDOC for supporting this project.

7. REFERENCES

[1] Ahanger, G., Little, T. D. C.; *A Survey of Technologies for Parsing and Indexing Digital Video*; Journal of Visual Communication and Image Representation, (Special Issue on Digital Libraries), March 1996, Vol. 7, No. 1, pp. 28-43

[2] Andrès, F., Ono, K.; *Active Hypermedia Delivery System*; Proc. of the Int'l Conf. On Data Engineering (ICDE'98), Florida, February 1998, pp. 600.

[3] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.; *Pattern-oriented Software Architecture – A System of Patterns*; Wiley and Sons, 1996.

[4] Cattel, R. G. G., Barry, D., Bartels, D., Berler, M., Eastman, J., Gamerman, S., Jordan, D., Springer, A., Strickland, H., Wade, D.; *The Object Database Standard: ODMG 2.0*; Morgan Kaufmann Publishers, Inc., San Francisco, California, 1997, 270 p.

[5] Chang, S., Smith, J., Beigi, M., Benitez, A.; *Visual Information Retrieval from Large Distribued Online Repositories*; Communications of the ACM, Vol. 40, No. 12, December 1997, pp. 63-71.

[6] Frakes, W. B., Baeza-Yates, R.; *Information Retrieval: Data Structures & Algorithms*; Prentice-Hall, 1992, 504 p.

[7] Goldberg, D. E.; *Genetic Algorithms*; Addison-Wesley, 1991

[8] Lavender, R. G., Schmidt, D. C.; *Active Object: an Object Behavioral Pattern for Concurrent Programming*; in *Pattern Languages of Program Design*, Reading, MA : Addison-Wesley, 1996.

[9] Martinez, J., Guillaume, S.; *Colour Image Retrieval Fitted to Classical Querying*; Proc. of the 9th Int'l Conf. on Image Analysis and Processing (ICIAP'97), Vol. II, Florence, Italy, September 17-19, 1997, pp. 14-21 (in LNCS No. 1311).

[10] Martinez, J., Guillaume, S.; *Colour Image Retrieval Fitted to Classical Querying*; Networking and Information Systems Journal, Vol. 1, No 2-3, 1998, pp. 251-278 (Editions HERMES, Paris, ISBN 2-86601-731-5).

[11] Martinez, J., Marchand, S.; *Towards Intelligent Retrieval in Image Databases*; Proc. of the 5th Int'l Worshop on Multi-Media Data Base Systems (MMDBMS'98), August 5-7, 1998, Dayton, Ohio, USA, pp. 38-45 (IEEE Computer Press, Los Alamitos, California, USA, ISBN 0-8186-8676-6)

[12] Martinez, J., Mouaddib, N.; *Multimedia and Databases: A Survey*; Networking and Information Systems Journal (NISJ), Vol. 1, No. 6, 1999 (Editions HERMES, Paris)

[13] Michalski, R. S., Bratko, I., Kubat, M. (Eds.); *Machine Learning and Data Mining*: Methods and Applications; John Wiley & Sons, New York, 1998, 456 p.

[14] Mitchell, T. M.; *Machine Learning*; WCB/MacGraw-Hill, 1997, 414 p.

[15] Ogle, V. E., Stonebraker, M.; *Chabot: Retrieval from a Relational Database of Images*; IEEE Computer, September 1995, pp. 40-48.

[16] Santini, S., Jain, R.; *Beyond Query by Example*; Proc. of the 6th Int'l ACM Conf. on Multimedia (MM'98), Bristol, UK, September 1998.

[17] Schmidt, D. C., Suda, T.; *An Object-oriented Framework for Dynamically Configuring Extensible Distributed Communication Systems*; IEE/BCS Distributed Systems Engineering Journal (Special Issue on Configurable Distributed Systems), Vol.2, December 1994, pp. 280-293.

[18] Swain, M. J., Ballard, D. H.; *Color Indexing*; International Journal of Computer Vision, 7(1), 1991.

[19] Swain M. J.; *Interactive Indexing into Image Databases*; Proc. of SPIE, Storage and Retrieval for Image and Video Databases, Vol. 1908, San Jose, California, February 1993, pp. 95-103.

[20] Java ACE Home Page. The ADAPTIVE Communication Environment in Java version 1.4.6, Washington University/CDOC. www.cs.wustl.edu/schmidt/JACE.html