PICSearch – A Platform for Image Content-based Searching Algorithms

Kjell Lemström *

University of Helsinki
Department of Computer Science
P.O. Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki
Finland
klemstro@cs.helsinki.fi

Jouni Korte, Pyry Kuusi, Pasi Kyheröinen, Pekka Päiväkumpu

Helsinki University of Technology
Department of Computer Science
Otakaari 1
FIN-02150 Espoo
Finland
{jones,pkuusi,pky,ppaivaku}@cc.hut.fi

Abstract

An environment for content-based pictorial retrieval algorithms, called **PICSearch**, is introduced. In this context, the retrieval is based on color distribution, texture or edges of a *query image*, or a *sketch*; i.e. the boundary information of a shape. **PICSearch** is designed to serve as a platform for any kind of pictorial matching algorithm. The system is designed for researchers developing such algorithms. **PICSearch** provides an easy-to-use graphical interface and a platform, where the researcher can easily embed his algorithm without the need to create a whole system from scratch. **PICSearch** is very independent on the underlying operating system and window manager. **PICSearch** is released to public use (under the GNU General Public License) and, to our knowledge, it is the first open platform to image retrieval systems freely available.

Keywords: content-based image retrieval, visual information management, open platforms, image databases

Computing Reviews (1991) Categories:

H.3.3	[Inf. Systems]: Inf. Storage and Retrieval – Information Search and Retrieval
I.4.9	[Comp. Methodologies]: Image Processing – Applications
I.5.5	[Comp. Methodologies]: Pattern Recognition – Implementation

1 INTRODUCTION

Image indexing, storage and retrieval based on pictorial data is becoming an ever more important issue because of many application domains and increasing computational power of computers. Many image retrieval systems have so far been based on associating text or keywords to images. This method has many drawbacks. An exhaustive set of keywords that describes the content of an image is impossible to generate, because even two persons tend to select totally different keywords. There are also images that are very difficult to describe using only textual information. Recently, some projects have concentrated on pattern recognition of pictorial data without any textual information.

In a system where the retrieval of images is based on the content, the queries are actualized comparing features of the images instead of textual descriptions of the features. These features can be colors, textures or edges of an image. Sketching provides an interesting query method. In such a query the occurences of a given shape are tried to be found in the database. We encourage the reader to become acquainted with some existing methods [Pet93, JFS95, Gra95, JZL96].

In an image retrieval system several components are needed. A *segmentation* component is needed if one wants to retrieve an object that is only a part of an image. The images in a database are segmented and the segments (the interesting parts of the images) can be stored in an *index structure*. This structure is used to avoid the linear behavior of the search process. An important component is also an *image editor* which can be used to create or edit a query image. For the database usage a *database engine* is needed. Then, of course, one needs the *matching* or *recognition* algorithms and the corresponding *similarity* or *distance measures*. Eventually, if one wants to expand the database, a *search agent* is of use to find more images to the database, for instance, over the Internet.

In this paper we introduce a system called **PICSearch**, which can be used as a platform for different image content-based searching algorithms. Many researchers are currently working in the area of pictorial pattern recognition. The primary aim of this system is to support the work of a researchers by offering them a platform system that is freely available, can be used

*The financial support of the Helsinki Graduate School of Computer Science and Engineering is appreciated.

in different environments and is not bound to any particular database management system. By using **PICSearch** they can concentrate on developing and testing new retrieval algorithms without bothering to create a whole system from scratch. The system is intended to be an ever expanding toolbox. Later, when various different searching algorithms are embedded in the system, it provides a retrieval tool also for "normal" use.

The main emphasis of the system has been on the architecture. The architecture has to be modular and the *APIs* (Application Programming Interfaces) have to be well defined to allow easy embedding of different methods needed in a pictorial retrieval system. Other important areas of concentration have been the graphical user interface and the portability of the system.

The rest of this paper is organized as follows: In the next Section we will review other content-based image retrieval systems. Section 3 gives an overview to our system. Section 4 describes how **PICSearch** is implemented and in the succeeding Section 5 we will give a sketched example of embedding an algorithm into the system to give an idea how easily this can be done. In Section 6 we report some limitations of the current system. The future tasks are presented in Section 7 and we conclude the paper in Section 8.

2 RELATED WORK

In conventional image databases, image retrieval is achieved by using associated textual information, whereas the modern image retrieval systems use sophisticated retrieval methods based directly on the features of the image. Navigation in an image database can be made more effective by supporting combination of visual information and textual annotations. For instance, *Piction* [Sri95] is a content-based photograph retrieval system, in which the fusion of newspaper photographs and their captions is exploited.

The projects most similar to **PICSearch** are *QBIC*, *Virage* and I^2C. All these enable the addition of some custom defined components. The QBIC[1] (Query By Image Content) system developed by Niblack et al. [FSN95, LBN94a, LBN94b, FBF94] is a project of IBM. QBIC offers several query options: query by color, by example, by sketch, by texture or on the position of the objects that the query image contains. These attributes can be combined to a multi-feature query. QBIC technology can be combined with a commercial product, called *Ultimedia Manager* [TOH93]. This is a tool to prepare images for a content-based query: the interesting parts of an image can be identified. The system computes their features.

The Virage Engine[2] [BFG96] is based on the *VIMSYS* model [GWJ91]. Image features considered in Virage include texture, color and structure. These are called as primitives and they can be both global and local. The primitives of the entire image and the regional properties of the image are extracted by different processes and every computational process is assigned with a corresponding distance metrics. The user can adjust a set of weighting factors that are assigned to the different distance metrices. The resulting composite metrics is used to compare the similarity between the query image and the database images. Virage technology has been combined with an object relational database called *Illustra*.

I^2C is a system developed by Orphanoudakis, Chronaki and Kostomanolakis [OCK94] for medical image indexing, storage and retrieval. In I^2C the user can embed new description types, segmentation algorithms, image processing algorithms and properties. The architecture of **PICSearch** is based on the architectural ideas of I^2C. They provide a classification-tree to browse/edit. Different attributes can be assigned to different classes. Orphanoudakis et al. [OCV96] have continued their work by implementing I^2Cnet. It is a network of servers that provides content-based query services through a WWW server.

The IRIS [AHK96] system (Image retrieval for images and videos) offers the possibility to retrieve an image out of a video stream. The preprocessing of a video stream consists of two steps. The first one is to extract shots from the stream. The shots consist of images having features (color histograms) of the same kind. The second step, which is called *mosaicing*, is to combine the images of a shot to a single larger image.

Van den Berg et al. [BBW96] have concentrated on integrating image processing and database paradigms. Like we, they also try to facilitate the construction of an image retrieval system. Their approach is different: they offer a detector based image processing module integrated with an object-oriented database. The database they use is based on Monet> database [BeH96].

Djeraba et al. [DSBM96] introduce a concept, in which the image features (colors, textures and shapes) are extracted automatically and some other features, such as keywords and relevant regions, can be extracted manually. The automatically and manually extracted features are represented in an object-oriented database.

Furthermore, there is a multimedia retrieval project called MARS[3] (Multimedia Analysis and Retrieval System) going on at the Beckman Institute of University of Illinois. The description of properties of MARS can be found e.g. in [MROH97].

[1] The demo of QBIC can be found for the present at: http://wwwqbic.almaden.ibm.com/cgi-bin/QbicStable.

[2] The demo of Virage can be found for the present at: http://206.169.1.90/cgi-bin/query-c.

[3] The demo of MARS can be found for the present at: http://quark.ifp.uiuc.edu:2020/marsApplet.html.

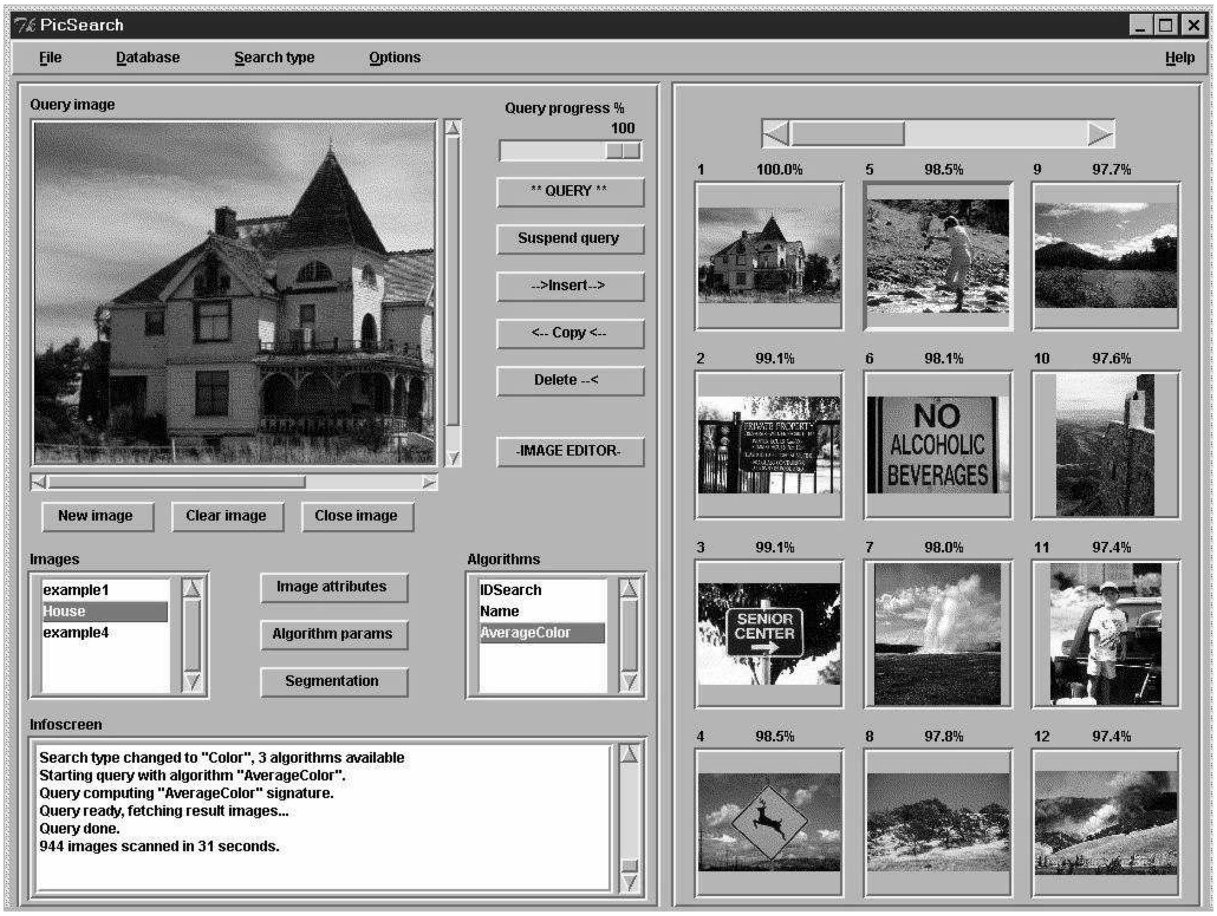

Figure 1: *The user interface of* **PICSearch**.

3 OVERVIEW OF PICSearch

PICSearch is a platform designed for content-based image retrieval providing a simple way of adding new algorithms. However, adding new retrieval algorithms is not the only way to extend **PICSearch**. Segmentation and other image processing tools can also be added. An existing image database engine could also be integrated to the system. In the sequel, we call a person who uses our system by adding new algorithms a *maintainer* (to distinguish programmers of the system and algorithm programmers).

The basic method for queries is based on algorithms that can support *signatures*. A signature is a predefined presentation computed out of an image and it is to be used in comparison to determine the similarity of two images. Thus in this scheme the algorithms must be able to supply a small description of each image and determine similarity by using only these descriptions. A database stores images and their signatures, which are always computed when an image is added to the database. In this way the comparison can be made efficiently since there is no need to compute the signatures of the database images at the time of the comparison.

PICSearch is programmed mostly in C++ and follows the principles of object-orientation. The user interface was constructed with script language Tcl/Tk. By using these tools and restricting operating system dependent code into as small areas as possible, we were able to make **PICSearch** an easily portable system. So far we have run the system using *Linux*, *Digital UNIX*, *Windows95* and *Windows NT*.

At the moment only a small set of algorithms is provided: searching by *id* (picture identifier), by associated keywords, by average color and a Wavelet-based matching algorithm [JFS95]. By providing these algorithms we were able to test the system and assure ourselves that it works correctly.

3.1 Graphical User Interface

The graphical user interface (*GUI*) provides access to all functionality the system includes. Modifications to the user interface are very easy to make and require no recompilation of the system. The graphical interface of **PICSearch** is shown in Figure 1.

Query Image. The queries are almost always based on a query image. This image is meant to represent the target the user is searching for. **PICSearch** enables the user to have a small set of query images opened and quickly switch between them. Each query image can be loaded and saved to a file, copied from the database and modified using a separate editor window.

Querying. After having supplied a query image the user may select a desired search algorithm from the list shown on the middle of the main window. For convenience, search algorithms are grouped into search types and only the algorithms registered under the type chosen in the search type menu are shown. The actual query is launched by the "query" button in the main window or in the image editor. Since each algorithm may have some parameters associated to the algorithm, these can be accessed by opening a window defined by a maintainer. The parameters are supposed to affect only the comparison method, not the way signatures are computed since they are already in the database.

On the right side of the main window are located the frames used to show the query result images. A scrollbar above the frames can be used to browse through the images. On top of each image, a value representing the similarity between the query image and the database image, is shown. The actual meaning of this value depends on the algorithm used in the query.

Database Manipulation. Images can be added to the database in a batch by typing in a file name mask or supplying a file containing the names of the image files to be inserted. The user may also insert the active query image into database pressing a button in the main window. A selected result image can be made a query image or deleted from the database.

Provisions for the Maintainer. By using a separate window, the user may set the name and other parameters of a query image. In the same window a maintainer-given textual representation of each signature of the image can be viewed. A small part of the main window is reserved for displaying messages to the user and debug information to the programmer.

The Image Editor. One central part of the interface is a separate drawing window with drawing tools (see Figure 2). This window is opened when the user wants to create a new query image or manipulate an old one. Several general drawing tools are included. Also a manner to apply an image processing algorithm to the image being edited are provided. Image processing algorithms are separate from searching algorithms: they manipulate images instead of computing signatures.

4 IMPLEMENTATION

In this section we will give a description of the implementation issues of **PICSearch**. The system is coded using C++, but the user interface was constructed with script language Tcl/Tk (versions 7.5 and 4.1).

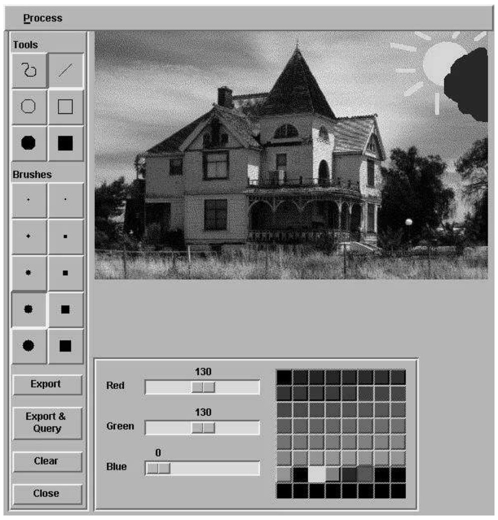

Figure 2: *The image editor of* **PICSearch**.

First, there is a general class library **PicLib** which implements all the basic classes used throughout the system. These include classes for string manipulation, generic arrays and operation system dependent features such as file handling and thread control.

The second subsystem **PicCore** implements the basic functionality of **PICSearch**. This main module processes all the messages received from the user interface and uses the database and the algorithms available to perform the actions requested.

The third subsystem **PicGUI** implements the user interface. All the user interface issues are wrapped inside this module and the main module completely ignores the visual aspects of the system. The user interface module also includes some functionality, which was easy to implement with the tool used and seemed of little importance to the main module.

The fourth module **PicAlg** consists of the algorithm and image handling functionality. The main module uses blindly the algorithms compiled into the system. The algorithms are accessed through an API. The same approach applies to the storage formats of images.

Finally, the implementation of permanent storage for the data is wrapped into **PicDb** module. This subsystem provides an interface for the main module to write and read images, signatures, etc. So far, this subsystem includes a straightforward database implementation. The main module does not know how this module operates. Therefore, the incorporation of a more advanced database engine should be straightforward. In current implementation of **PICSearch** we store into the database the following information of each image:

- Id (used as key), name, height, width, image format identifier and a link to the actual image (text string) or actual image data. Furthermore, a small thumbnail version of each image is maintained, as well as two user given parameters.

- Arbitrary-length signatures computed by the search algorithms available in the system.
- Segmentations consisting of arbitrary-length segments computed by the segmentation algorithms available in the system.

Information about algorithms and system parameters are also kept in the database.

4.1 Core Module

The basic functionality of **PICSearch** is mostly implemented in the main module. However, the actual program execution is usually inside the GUI module and the main program is only called when the user performs some action. Some of the long operations are executed concurrently as separate threads. For example, queries are run as threads and therefore the user interface does not freeze during the query operation.

The GUI module accesses the main program through a well-defined set of functions, one for each action the user may perform. As the main program proceeds it gives the possible results back to the GUI or starts a suitable thread and returns immediately. Ideally, the system could be separated from Tcl/Tk and replaced by another GUI mechanism, but we see no reason for doing so, except for the porting to an interface that can be used by an Internet browser.

At system start-up the main program connects to the database, starts GUI and provides GUI with the names of available algorithms and search types. If the system detects algorithms that are not specified in the database, it assumes that these algorithms were compiled into the system since last start-up and launches a thread to update the database.

4.2 User Interface

The user interface module consists of all code related to the GUI. Major part of this module is made up of Tcl/Tk scripts, which implement the actual windows and the immediate functionality behind every screen component. A minor part of this module consists of C++ code needed to make the connection between this module and the main program.

The Tcl/Tk scripts need to be extended when adding a new algorithm to the system if the algorithm has some parameters affecting the matching function. For example, a color matching algorithm might have three parameters controlling which color components to use in matching during a query. The same scheme applies to image processing algorithms, which may freely define the parameters they need.

Since the user interface uses Tcl/Tk scripts, which are interpreted at run-time, the changes in the GUI require no recompilation of **PICSearch**.

4.3 Database Module

PicDb subsystem offers methods to access the **PICSearch** database, which stores the collection of images and associated information to be used in queries. To increase flexibility, only a "link" to the image actually residing, for example, somewhere on a disk can be stored in the database instead of the actual image data. Furthermore, some system parameters requiring non-volatile storage are kept in the database.

All communication between the core of **PICSearch** and **PicDb** is performed through the database interface, **PicDb-API**. It covers standard database access services: connecting to the database, disconnecting, submitting a query and fetching the results of a query, inserting, deleting and updating information in the database, and commit/rollback operations.

PicDb-API was designed to be as straightforward as possible to be implemented using an underlying, preferably relational, Database Management System (DBMS). A short code is needed to map **PicDb-API** functions to corresponding functions of the DBMS-API. The database used must be able to store values of arbitrary length (images, signatures, segments). Hence, a basic relational database cannot be used.

Currently, the functionality behind **PicDb-API** is implemented using **PicDbEngine**. It serves well with a relatively small test database when using **PICSearch** as a test platform for new image search algorithms. When optimal performance or reliability is needed a more sophisticated DBMS should be used.

PicDbEngine is implemented with simplicity in mind. Relational tables are all saved in their own files using services of the operating system. There are two files per relation table: the actual data file containing the table entries and a dense index file. There is an entry in the index file for every entry in the data file.

4.4 Algorithm Module

The algorithm module gives an easy way to embed new algorithms. New algorithms can be added by deriving a new C++ class from a common base class redefining a couple of functions and recompiling the system.

In Table 1 the first two functions, *getParams* and *setParams*, in the first class are used to access the possible parameters of the matching function. The function *compare* is called to compute the similarity between two images represented by the argument signatures. The last function of the first class, *computeSignature*, is called to compute a signature out of an image. In this case, the return value would most likely be an instance of class *MySignature*, which includes methods for generating a suitable binary representation of the signature to be inserted in the database. Also a method for regenerating the signature out of the binary data read from the database is needed.

```
class MySearchAlg : public PicSearchAlgorithm
{ public:
  ...
  virtual PICBOOL getParams(PicIxCollect &params);
  virtual PICBOOL setParams(PicIxCollect &params);
  virtual PICUSHORT compare(PicSignature &firstSig, PicSignature, &secSig);
  virtual PicSignature *computeSignature(PicPicture &pic);
  ...
};

class MySignature : public PicSignature
{ public:
  ...
  virtual char *getBin(PICULONG &size) const;
  virtual PICBOOL setBin(const char *data, PICULONG size);
  ...
};
```

Table 1: *Embedding an algorithm into* **PICSearch**.

```
PicSignature *PicAvgColorAlgorithm::computeSignature(PicPicture &pic)
{
  int size = pic.raw->getWidth() * pic.raw->getHeight();
  for(ix=0; ix<3*size; ix += 3){
    redSum += data[ix]; greenSum += data[ix+1]; blueSum += data[ix+2];
  }
  return PicAverageColorSignature(redSum/size, greenSum/size, blueSum/size);
}
```

Table 2: *Implementing the signature of the average-color algorithm.*

A programmer who is familiar with the programming environment should be able to implement algorithms without excessive difficulty. In the next Section we will give a sketch how a simple example algorithm is embedded into the system.

The algorithm module also includes similar classes for segmentation and image processing algorithms. Furthermore, a similar scheme is used for image formats (i.e. *JPEG*, *PPM* and *Sun Raster*).

5 EMBEDDING ALGORITHMS

Adding a new algorithm requires little knowledge about the inner workings of the system. Copying two files under new names (See Table 1) and filling in non-trivial code for six functions is enough. For a complex algorithm such as the Wavelet-based matching algorithm [JFS95], the following procedure is analogous, but the computation is more involved. For a simple matching algorithm based on the average color of images, the following steps were taken:

1. The signature was designed to be fixed-length (3 bytes), thus one variable for the average value of each color component were added to the *Signature*-class implementation.
2. Writing and reading these values to/in a memory buffer was trivial (functions *getBin* and *setBin*). Generating a human-readable representation of the signature was equally trivial.
3. The algorithm was designed to use three parameters, each controlling whether to use a corresponding color component during a query. Hence one variable for each parameter was added to the *Algorithm*-class implementation. Straightforward code for reading/writing the paramaters was added (functions *getParams* and *setParams*).
4. A new window was created (requiring only about 15 lines of Tcl/Tk-code) in the GUI in order to control the parameters.
5. The actual computation of the signature was coded (See Table 2): a simple loop summing up the color component values from a given memory buffer and finally dividing them by the size of the image and setting the values into a new *Signature*-class instance.
6. Finally, the method for comparing two signatures was coded to interpret the color components as three-dimensional co-ordinates. The distance between two given signatures is calculated out of this representation and the solution is scaled to the required interval (0-100) (See Table 3).

To test the algorithm, the system was recompiled and the database automatically processed through using the new algorithm. An experienced maintainer implemented the algorithm in 2 hours.

```
PICUSHORT PicAvgClrAlgorithm::compare(PicSignature &firstSig, PicSignature &secSig)
{
  if(useRed && useGreen && useBlue){
    diff = (firstSig.avgRed - secSig.avgRed);
    sum += diff * diff;
    ...
    return 100 - (sqrt(sum) * SCALE_FACTOR);  // scaled to 0-100
  }
}
```

Table 3: *Implementing the function that compares average-color signatures.*

6 LIMITATIONS

The basic limitation of **PICSearch** is that the algorithms of the maintainers have to fit into the model used: each must be able to produce a signature out of the image data and be able to give a match rate (scaled to a common range) between two signatures. If an algorithm cannot satisfy these restrictions the query procedure has to be modified.

In the current system the usage of segmentation was left open for future development. Since there seems to be no single method that could be forced on all segmentation based algorithms, we did not enforce one. The segmentations as well as signatures of images are computed automatically when an image is inserted into the database. The query process needs to be augmented to enable a suitable usage of the segmentations.

In the current *Windows* version of our system only modes up to 256-colors can be used. This is because in Tcl/Tk it is not possible to conveniently draw graphics into bitmaps. However, graphic objects can easily be drawn on window canvas. Because of this limitation of Tcl/Tk if the query image is edited in the Image Editor it has to be exported to the main window by using some trick. We grab the image directly from the screen and this operation does not yet work properly in *Windows*.

For the present it is not convinient to delete an existing algorithm from the system. This causes that the batch insertion for a great number of images is a relative slow operation because a signature for every image and for every algorithm has to be calculated. This process could be speeded up if the unnecessary algorithms could be exluded. However, this is not a too uncomfortable property because the signature calculation is only done when an image is inserted to the database and the massive insertion is an infrequent operation.

7 FUTURE TASKS

PICSearch was developed to be an open platform for adding different algorithms used in queries based on image content. Thus, the most obvious way to develop the system is to add new algorithms to it.

The queries are currently based on one query image only. A modification which enables the use of multiple query images would be practical in some situations. Enabling the use of multiple algorithms at the same time would, however, require rethinking of the user interface and query method. To enable efficient queries from a large database the system has to be combined with a "real" database engine and an efficient index structure has to be used. However, we do not want to bind **PICSearch** to any DBMS because of the portability aspect. The development and implementation of an efficient index structure is left as an open issue.

We have an on-going project whose aim is to develop a client-server version of the current system. The databases that can be scanned, have not to be any more local. Instead, if one has the client version of **PICSearch** he can scan databases that comply with a certain (server) protocol, through the Internet. The project tries also to solve some of the problems reported in the previous section.

8 CONCLUSIONS

We offer a platform for researchers working in the area of pictorial data retrieval. **PICSearch** is developed to facilitate the work of researchers so that they can concentrate on developing algorithms and implementing tools without the bother of creating a whole retrieval system. The contribution of **PICSearch** is that it is free, extendible and portable. This should be a combination appreciated by the researchers. Moreover, **PICSearch** is the first platform to an image retrieval project freely available, to our knowledge.

The embedding of segmentation, image processing, matching and searching algorithms should be straightforward by using **PICSearch**. **PICSearch** can be combined with an existing database engine. We offer straightforward routines to those who do not want to connect **PICSearch** to any particular database engine.

PICSearch is independent of the underlying operating system and window manager. It is implemented by using C++ and Tcl/Tk. The modularity and object-oriented approach facilitate the embedding of algorithms to the system. We have run **PICSearch** using *Windows95*, *Windows NT*, *Linux* and *Digital UNIX*.

Acknowledgements and Notes

We thank professor Tapio Takala for his guidance and comments during the project. We also thank professor Esko Ukkonen and Oskari Heinonen for their comments on this paper. Everything associated with the system can be found in the homepage of **PICSearch**: *http://www.cs.Helsinki.fi/~klemstro/picsearch.html*. **PICSearch** is intended to be an ever expanding toolkit. If you regard the system as an useful software and embed some algorithm(s) or tool(s) into the system, please contact us for public release of the algoritm(s).

References

[AHK96] P. Alshuth, T. Hermes, C. Klauck, J. Kreyß, and M. Röper. Iris: Image retrieval for images and videos. In *Proceedings of the First Int. Workshop of Image Databases and Multi-Media Search*, pages 170–178, 1996.

[BBW96] C. A. van den Berg, R. van den Boomgaard, M. Worring, D. Koelma, and A. Smeulders. Horus: Integration of image processing and database paradigms. In *Proceedings of the First Int. Workshop of Image Databases and Multi-Media Search*, pages 226–234, 1996.

[BeH96] C. A. van den Berg and A. van der Hoeven. Monet meets 007. In *Object-Oriented Database Systems Symposium*, 1996.

[BFG96] J. R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain, and C. Shu. The Virage image search engine: An open framework for image management. In *Proceedings of the SPIE: Storage and Retrieval for Still Image and Video Databases IV*, February 1996.

[DSBM96] C. Djeraba, I. Savory, M. Barrere, and S. Marchand. Content based image retrieval model in an object oriented database. In *Proceedings of the First Int. Workshop of Image Databases and Multi-Media Search*, pages 187–193, 1996.

[FBF94] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, (3):1–28, 1994.

[FSN95] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32, 1995.

[Gra95] R. S. Gray. Content-based image retrieval: color and edges. Technical Report TR95-252, Department of Computer Science, Dartmouth College, USA, 1995.

[GWJ91] A. Gupta, T. Weymouth, and R. Jain. Semantic queries with pictures. In *Proceedings of the 17th International Conference of VLDB*, pages 69–79, 1991.

[JFS95] C. E. Jacobs, A. Finkelstein, and D. H. Salesin. Fast multiresolution image querying. In *SIGGRAPH'95*, pages 277–286, 1995.

[JZL96] A. K. Jain, Y. Zhong, and S. Lakshmanan. Object matching using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(3):267–277, March 1996.

[LBN94a] D. Lee, R. Barber, W. Niblack, M. Flickner, J. Hafner, and D. Petkovic. Indexing for complex queries on a query-by-content database. In *IAPR International Conference on Pattern Recognition*, pages 142–146, October 1994.

[LBN94b] D. Lee, R. Barber, W. Niblack, M. Flickner, J. Hafner, and D. Petkovic. Query by image content using multiple objects and multiple features: User interface issues. In *International Conference on Image Processing*, 1994.

[MROH97] S. Mehrotra, Y. Rui, M. Ortega, and T. S. Huang. Supporting content-based queries over images in MARS. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems ICMCS'97*, June 1997.

[OCK94] S. C. Orphanoudakis, C. Chronaki, and S. Kostomanolakis. I^2C: A system for the indexing, storage, and retrieval of medical images by content. *Journal of medical information*, 19(2):109–122, March 1994.

[OCV96] S. C. Orphanoudakis, C. E. Chronaki, and D. Vamvaka. I^2Cnet: Content-based similarity search in geographically distributed repositories of medical images. Technical Report 161, Institute of Computer Science, Foundation for Research and Technology-Hellas, Crete, Greece, 1996.

[Pet93] E. G. M. Petrakis. *Image representation, indexing and retrieval based on spatial relationships and properties of objects*. PhD thesis, University of Crete, Department of Computer Science, March 1993.

[Sri95] R. K. Srihari. Automatic indexing and content-based retrieval of captioned images. *IEEE Computer*, 28(9):49–56, 1995.

[TOH93] H. Treat, E. Ort, J. Ho, M. Vo, J. Jang, L. Hall, F. Tung, and D. Petkovic. Searching images using Ultimedia manager. Technical report, IBM Santa Teresa Lab, 1993.