Pixel-Based Searching of Images Stored in a Database

Igor Stojanovic[1] and Momcilo Bogdanov[2]

Abstract – **In the paper we apply pixel-based searching of images stored in a database. We make use of progressive wavelet correlation along with Fourier methods. The searching consists of three incremental steps, each of which quadruples the number of correlation points. The process can be halted at any stage if the intermediate results indicate that the correlation will not result in a match.**

Keywords – **JPEG, database, wavelets.**

I. INTRODUCTION

A key tool that helped make the Internet universally useful is the text-search engine. The image-search engines available today are relatively crude. There are several techniques for image searching: descriptor-based search, pixel-based search and image understanding techniques. The fastest methods available today use descriptor-based search techniques. IBM QBIC (www.qbic.almaden.ibm.com) [1] is an example of this type of search engine. Images with higher information content, such as satellite images and medical images, are difficult to encapsulate with descriptors. Queries on images of this type require detailed analysis. Normalized correlation coefficients, an instance of pixel-based search techniques, measure the differences between images and patterns. They can be computed with progressive wavelet correlation using Fourier methods [2]. The images are mapped into the wavelet-frequency domain to take advantage of high-speed correlation.

The paper is organized as follows. Section II contains the brief description of progressive wavelet correlation using Fourier methods [2]. An implementation of progressive wavelet correlation using Fourier methods for searching of images stored in a database is presented in Section III.

II. STONE'S METHOD OF PROGRESSIVE WAVELET CORRELATION

A. Overview of the Method

The basic idea of Stone's method includes: elimination in the earlier phases of searching, correlation in the frequency domain for fast computation, DCT in factorizing form, and wavelet representation of signals for efficient compression [2].

The algorithm consists of three incremental steps:

1. *Coarse correlation* – every eighth point of the correlation is generated.
2. *Medium correlation* – obtain the correlation at indices that are multiples of 4 mod 8 of the full correlation.
3. *Fine correlation* – obtain the correlation at indices that are multiples of 2 mod 8 and 6 mod 8 of the full correlation.
4. *Full correlation* – obtain the correlation at odd indices.

Fig.1 is a flow diagram showing the steps performed for an image search according to the method.

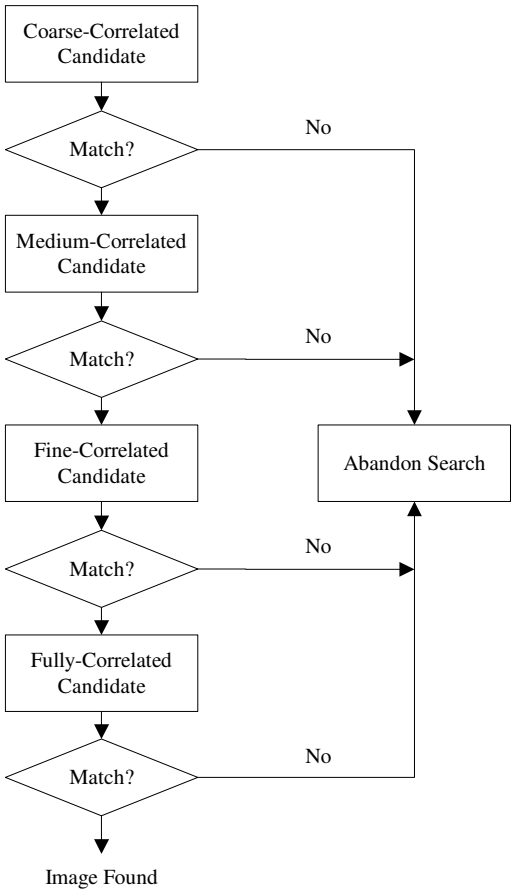


Fig. 1

There are two ways of using the method. The first one is for locating a given image in an image library, and the second one is for locating objects in the JPEG image [5]. Examples of locating a given image, locating an object in a JPEG image and conclusions are given in [3] [4].

[1]Igor Stojanovic is with the ICT Sector, Customs Administraton of RM, Lazar Licenoski 13, 1000 Skopje, Macedonia,
E-mail: igor.stojanovic@customs.gov.mk

[2]Momcilo Bogdanov is with the Faculty of Electrical Engineering, Karpos II b.b., P.O. Box 574, 1000 Skopje, Macedonia,
E-mail: bogdanov@ etf.ukim.edu.mk

B. Two-Dimensional Case

We investigate what happens in the two-dimensional case. The assumption is that the image size is N by N. In step 1, we have 64 subbands of length $N^2/64$. We perform one step of the inverse 2D **H** function, and one 2D step of the forward Fourier transform function. Fortunately, these steps are simple generalizations of the 1D functions. Specifically, if in 1D we compute **Hx** (JPEG transform of **x**), where **H** is N by N and **x** is N by 1, then in 2D we compute **H**2**X** conjtrans (**H**2), where **X** is the 2D image to be transformed, **H**2 is N by N, and each row of **H**2 is equal to **H**. This is equivalent to applying **H** to each of the columns of **X** and **H** to each of the rows of **X**. The next step is to add the 64 subbands point by point to create a 2D array of size $N/8$ by $N/8$. If we take its inverse Fourier transform, we will obtain the correlations at points that lie on a grid that is coarser than the original pixel grid by a factor of 8 in each dimension.

In step 2, we obtain 16 subbands of size $N^2/16$ by adding the 16 subbands point by point, and taking the Fourier inverse. We will obtain the correlation values on a grid that is coarser than the original grid by a factor of 4 in each dimension.

In step 3, we obtain 4 subbands of size $N^2/4$. In step 4, we obtain the full resolution.

Formulas for calculating normalized correlation coefficients that measure differences between images and patterns are given in [2]. Normalized correlation coefficients can be computed from the correlations described above. The normalization is very important because it allows for a threshold to be set. Such a threshold is independent of the encoding of the images.

The normalized correlation coefficient has a maximum absolute value of 1. Correlations that have absolute values above 0.9 are excellent, and almost always indicate a match found. Correlations of 0.7 are good matches. Correlations of 0.5 are usually fair or poor. Correlations of 0.3 or less are very poor. There is a tradeoff between the value of the threshold and the likelihood of finding a relevant match. Higher thresholds reduce the probability of finding something that is of interest, but they also reduce the probability of falsely matching something that is not of interest.

III. Adaptation of Stone's Method for Searching in a Database

A. Image Store and Matlab Database Toolbox

The method of Stone provides guidelines on how to locate an image in the image library. To make this method practical, we must first decide how to store the images. The initial choice is to store them in a disk file system. This can be seen as the quickest and simplest approach. Another, better alternative should be looked at and that is the move to store those images in a database. In the past five years, with changes in database technology and improvements in disk performance and storage, the rules have changed and it now makes business sense to use the database to store and manage all of an organizations' digital assets. The following are the strengths a database can offer over traditional file system storage: manageability, security, backup/recovery, extensibility, flexibility.

We use the Oracle Database for investigation purposes. There are two ways of image's storage into Oracle Database. The first one is the use of Large Objects – LOB, and the second one is the use of Oracle *inter*Media.

Unstructured data such as text, graphic images, still video clips, full motion video, and sound waveforms tends to be large in size. A typical employee record may be a few hundred bytes, while even small amounts of multimedia data can be thousands of times larger. Datatypes that are ideal for large amounts of unstructured binary data include the BLOB datatype (Binary Large Object) and the BFILE datatype (Binary File object).

Oracle *inter*Media is a feature that enables Oracle Database to store, manage, and retrieve images, audio, video, or other heterogeneous media data. *inter*Media uses object types, similar to Java or C++ classes, to describe multimedia data. These object types are called ORDAudio, ORDDoc, ORDImage, and ORDVideo. An instance of these object types consists of attributes, including metadata and the media data, and methods.

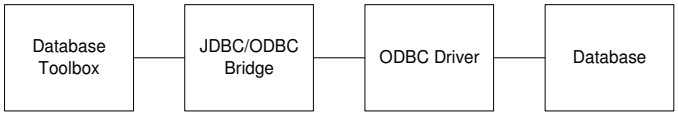
To store images into database we use the BLOB datatype. After creation of one BLOB column defined table we also create a PL/SQL package with loading of images procedure (load named) included. This procedure is used to store images into the database.

The implementation of the method of Stone into Matlab and connection of the algorithm with the database are next steps. The Database Toolbox is one of an extensive collection of toolboxes for use with Matlab. The Database Toolbox enables one to move data (both importing and exporting) between Matlab and popular relational databases. With the Database Toolbox, one can bring data from an existing database into Matlab, use any of the Matlab computational and analytic tools, and store the results back in the database or in another database. The Database Toolbox connects Matlab to a database using Matlab functions. Data can be retreived from the database and store it in the Matlab workspace. At that point, the extensive set of Matlab tools can be used to work with the data. Database Toolbox functions can be included in Matlab M-files. To export the data from Matlab to a database, Database Toolbox functions can be used. The Visual Query Builder (VQB), which comes with the Database Toolbox, is an easy-to-use graphical user interface (GUI) for exchanging data with your database. The VQB can be used instead of or in addition to Database Toolbox functions.

Before the Database Toolbox is connected to a database, a **data source** must be set. A data source consists of data that you want the toolbox to access, and information about how to find the data, such as driver, directory, server, or network names. Instructions for setting up a data source depend on the type of database driver, ODBC or JDBC.

For Windows platforms, the Database Toolbox supports Open Database Connectivity (ODBC) drivers as well as Java Database Connectivity (JDBC) drivers. For UNIX platforms, the Database Toolbox supports Java Database Connectivity (JDBC) drivers. An ODBC driver is a standard Windows

interface that enables communication between database management systems and SQL-based applications. A JDBC driver is a standard interface that enables communication between Java-based applications and database management systems. The Database Toolbox is a Java-based application. To connect the Database Toolbox to a database's ODBC driver, the toolbox uses a JDBC/ODBC bridge, which is supplied and automatically installed as part of the MATLAB JVM. The figure 2 illustrates the use of drivers with the Database Toolbox.



Windows platforms

UNIX and Windows platforms

Fig. 2

If the Windows-based database supports both ODBC and JDBC drivers, the JDBC drivers might provide better performance when accessing the database because the ODBC/JDBC bridge is not used.

The connection definition can be established using either the Oracle ODBC driver or the Microsoft ODBC driver for Oracle. To use these drivers, it is necessary to have Matlab and the Oracle client installed on the same computer. During testing we realized that Microsoft ODBC drivers for Oracle cannot be used for tables with columns of data type LOB.

For testing purposes JDBC drivers were usually used. If you have an Oracle database system that includes the JDBC drivers in the classes12.zip file, the following steps should be performed:

1. Create a directory $matlabroot/dbtools.
2. Put the classes12.zip file into dbtools.
3. Add the classes12.zip file to the classpath.txt file, by including this line in classpath.txt
4. $matlabroot/dbtools/classes12.zip
5. Restart MATLAB.

After setting up the data source for connecting to and importing data from a database we have used several standard functions of the Matlab Database Toolbox.

We can retrieve BINARY or OTHER Java SQL data types. However, the data might require additional processing once retrieved. For example, data can be retreived from a MAT-file or from an image file. Matlab cannot process these data types directly. One needs knowledge of the content and might need to massage the data in order to work with it in Matlab, such as stripping off leading entries added by the driver during data retrieval.

For purposes of saving of the extracted data into file "testfile", we created the Matlab file parsebin.m. Using the "imread" function, we stored the file date into a two-dimensional output variable x.

In working with the Microsoft Access database, ODBC drivers are used for extracting of the database data. The extracting process must contain adjustment in cutting of the header created by the driver. Let "m" be quantity of bytes attached to the beginning of the data package by the ODBC driver. The quantity of bytes depends on the file type (file extension).

To discovering the value of "m" we created a procedure in Matlab. That procedure helps us find the adequate value for "m" when linking with the image format.

When working with Oracle databases and extracting data with JDBC and ODBC drivers there is no need for adjustment, so "m" has always the value "1" (m="1").

B. HTTP Application

The last step in adaptation is to create Matlab applications that use the capabilities of the World Wide Web to send data to Matlab for computation and to display the results in a Web browser. The Matlab Web Server depends upon TCP/IP networking for transmission of data between the client system and Matlab.

In the simplest configuration, a Web browser runs on your client workstation, while Matlab, the Matlab Web Server (matlabserver), and the Web server daemon (httpd) run on another machine as shown in figure 3.

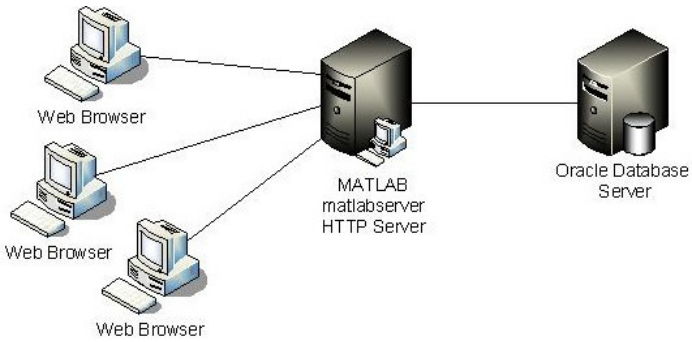


Fig. 3

In a more complex network, the Web server daemon can run on a machine apart from the others (Fig. 4).

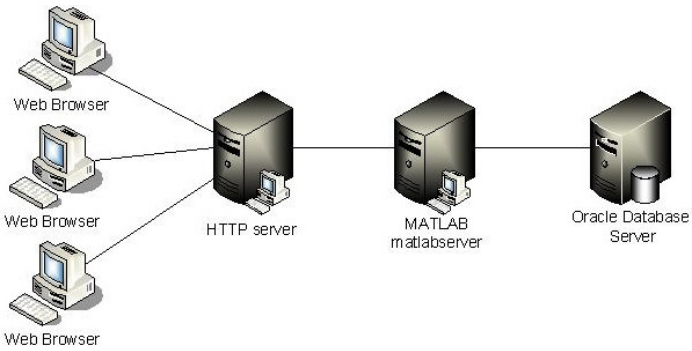


Fig. 4

Matlab Web Server applications are a combination of M-files, Hypertext Markup Language (HTML), and graphics.

The application development process requires a small number of simple steps:

1. Create the HTML documents for collection of the input data from users and display of output. You can code the input documents using a text editor to input HTML directly, or you can use one of the commercially available HTML authoring systems, such as Front Page from Microsoft etc.
2. List the application name and associated configuration data in the configuration file matweb.conf.
3. Write a Matlab M-file that:
   3.1. Receives the data entered in the HTML input form.
   3.2. Analyzes the data and generates any requested graphics.
   3.3. Places the output data into a Matlab structure.

The input mask of our application consists of three parameters: the image size *N*, the threshold lim, and the name of the image that we are looking for (Fig. 5).

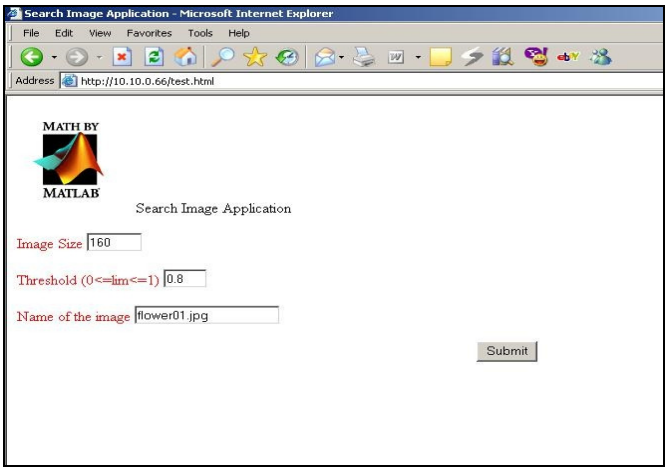


Fig. 5

IV. Conclusion

In this paper we presented our work that examines the pixel-based searching of images stored in a database. We exploited the technique of progressive wavelet correlation using Fourier methods, which led us to the following conclusions.

This technique is not yet suitable for general practical commercial usage. The reason for that is the big number of operations per picture. In the following years, with increasing processor speed, there should be a possibility for detail analysis at the rate of 1000 pictures per second. With this processing speed, it would be easy to construct a system, which is a combination of searching by description (descriptor-based search) and searching by pixels (pixel-based search). The descriptor can be used for isolating a certain part out of a big collection, which should be an object of a detailed pixel-based search in the later phase.

References

[1] M. Flickner et al., "Query by image and video content: The QBIC system," IEEE Comp., vol. 28, pp. 23-32, Sept.1995.
[2] H. S. Stone, "Progressive Wavelet Correlation Using Fourier Methods," IEEE Trans. Signal Processing, vol. 47, pp. 97-107, Jan. 1999.
[3] I. Stojanovic, M. Bogdanov, "Location of Objects in a JPEG Image with Progressive Wavelet Correlation using Fourier methods", 9th Telecommunications Forum, pp. 561-564, Belgrade, Yugoslavia, Nov. 20-22, 2001.
[4] I. Stojanovic, D. Taskovski, I. Kraljevski,"Normalized Correlation Coefficients for Searching JPEG Images", 7th Information Technologies 2002, pp. 104-107, Zabljak, Yugoslavia, 24 Feb. - 2 Mar., 2002
[5] G. K. Wallace, "The JPEG still-picture compression standard", Commun. ACM, vol. 34, no.4, pp. 30-44, Apr. 1991.