

# Online image database with content searching capabilities\*

Scott Alexander, Joshua Strohm, Denis Popel

Computer Science Department,  
Baker University  
Baldwin City, KS  
`denis.popel@bakeru.edu`

## Abstract

This paper outlines the research done to create a searchable image database in which image content is used rather than keywords. The outcome of this research is a database of images, which has user-friendly interface and numerous search capabilities in identifying individual images based on the image content alone.

---

\*This project has been done in cooperation with neoTROPY LLC, Lawrence, KS

# 1 Introduction

Image databases are currently considered to be a part of multimedia databases, and they differ from traditional information databases, which are easy to query, given that text is the object being searched for. Currently, there are no image databases which use an image itself as an object for a query. This is because a text-based query is computationally more efficient to perform than an image analysis of every image in the database. As a result, search engines such as *Google Image Search* actually query the title of the image rather than the content of the image itself. For example, a search for ‘duck’ will no doubt return some images of actual ducks, but it may also return images that are completely unrelated to the intended search. This is because such a search will return any image that has the text ‘duck’ in the title of the image (i.e. duck.jpg), instead of evaluating the content of the picture itself. An alternative method of circumventing this problem is to associate descriptive keywords with every image. Then a text-based query will return those images that have the appropriate associated keyword. The problem with this approach is that the descriptive keywords associated with the images are generated by a human, who will assign keywords based upon his/her interpretation of what is important to the image. Objectivity, in this case, is impossible to achieve since interpretation of images differs from one person to another. Not everyone is an expert at everything, and the fact remains that an expert can describe something better and more meaningfully than an amateur.

At first glance, objectivity does not seem hard to obtain. For instance, most people would properly classify a signature that spells ‘John Doe’ as belonging to the person John Doe. However, when asked for more specific keywords to describe a signature, we gain more objectivity. An average person might classify a signature as being ‘sloppy,’ whereas a signature expert might classify it by defining the curvature. These are both different, yet valid classifications of the same image. So for an image database in which specific objective distinctions are necessary for similar images, the keyword method of classification proves to be inadequate. Examples of images with such explicit requirements include fingerprints, signatures, and mug shots. This project is aimed at creating a solution for a database of signature images.

## 2 Solution

Since an image database of signatures based on keywords alone is inadequate for a qualitative analysis, we decided to implement a project that incorporates distinctions based on the data of the actual image. For instance, one might look at the image of a signature and deduce that it belongs to John Doe. Yet how would one differentiate the authentic signature from a forgery, as both signatures spell the name John Doe? It would be nearly impossible to accurately describe the two distinct images using keywords. However, the computer still needs to differentiate between the two images. A fairly simple computer analysis of the image

data can return many distinctive features. Using this analysis, we can ascertain features such as line thickness, curvature, pressure and other distinctive characteristics. These features are not only precise, but also completely objective and reproducible. This makes them desirable classifiers for an image database. Thus, we have found that content-based image searching and classification techniques are the best solution for this problem. It is also helpful to have an association of a keyword with an image for advanced functionality in searching. This, in turn, makes the program more robust. Some of the components of our solution include the server-based database (MySQL), different types of image acquisition tools (pressure sensitive tablets and flat scanners), image recognition and classification algorithms (C++ modules), and a web-based user interface (PHP modules).

## 3 Implementation

The first part of our implementation was to design the algorithms used to classify and compare the distinctive features of multiple images. This was accomplished by C++ modules which take image data, generate multiple features, and store them. This objective method of generating statistical data often solves the problem of differentiating between two different signatures of one name. While both signatures may spell the name John Doe, the two images may have vastly different curvature, speed, acceleration, and pressure features. Using these algorithms we can properly identify signature forgeries with a high degree of accuracy.

### 3.1 C++ modules

Image processing and classification algorithms have been implemented in a few proprietary C++ modules. These modules support the following functionality:

- generation of statistical data based on supplied images;
- comparison of two images providing a similarity score;
- generation of two-dimensional images for display purposes.

To facilitate the storing of this information, we decided to create a simple MySQL database containing information about the actual image, image type, associated keyword, and computer-generated characteristics derived from each image. Some of these derived features include curvature, speed, acceleration, and pressure.

### 3.2 MySQL database

The first step of our solution was to decide on the type and layout of our database [1]. We decided to create a database with three main tables: *Images*, *Statistics*, and *Results*.

Table 1: Database structure

| Attribute name      | Data type           | Description  |
|---------------------|---------------------|--|
| <i>Images</i>       |                     |  |
| <i>Id</i>           | integer             | unique id for all images   |
| <i>Label</i>        | character           | image description  |
| <i>Extension</i>    | character           | image file extension   |
| <i>File</i>         | long blob           | image binary data  |
| <i>File_type</i>    | character           | distinguishes between raster or vector images  |
| <i>Date_Added</i>   | date                | date image added to database   |
| <i>Statistics</i>   |                     |  |
| <i>Id</i>           | integer             | unique id for all images   |
| <i>Stat_type</i>    | character           | keyword describing the value vector (i.e. pressure or acceleration)                            |
| <i>Datatype</i>     | character           | data type used for the value vector (i.e. float or integer)                                    |
| <i>Value_vector</i> | long blob           | binary array of statistical data   |
| <i>Results</i>      |                     |  |
| <i>Id</i>           | integer             | unique id for all images   |
| <i>Similarity</i>   | float               | floating point number representing the similarity of two images, 0 being a perfect duplication |
| <i>Login</i>        |                     |  |
| <i>Username</i>     | character           | administrative username  |
| <i>Password</i>     | PASSWORD(Character) | password for the username hashed into a 16 byte encrypted hexadecimal string                   |
| <i>Last_Name</i>    | character           | last name of the user  |
| <i>First_Name</i>   | character           | first name of the user   |

Next we built the interface that allows users to execute and receive different queries. We decided to use a PHP web-based solution for this step as it is well suited to interact with our MySQL database [2; 3].

### 3.3 PHP modules

The PHP implementation has five basic functions: upload, list, compare, login, and add user. A successful login occurs after typing in a username and password combination that is found in the *Login* table of the MySQL database. After the successful login, the user is given administrative rights to the upload, list, compare and add user functions.

**Upload script** The upload script allows the user to upload an image providing a descriptive keyword to be saved in the *Label* field, and to choose a file type from the drop down list for the *File\_type* field (see Figure 1). After clicking submit, the PHP script inserts a new record

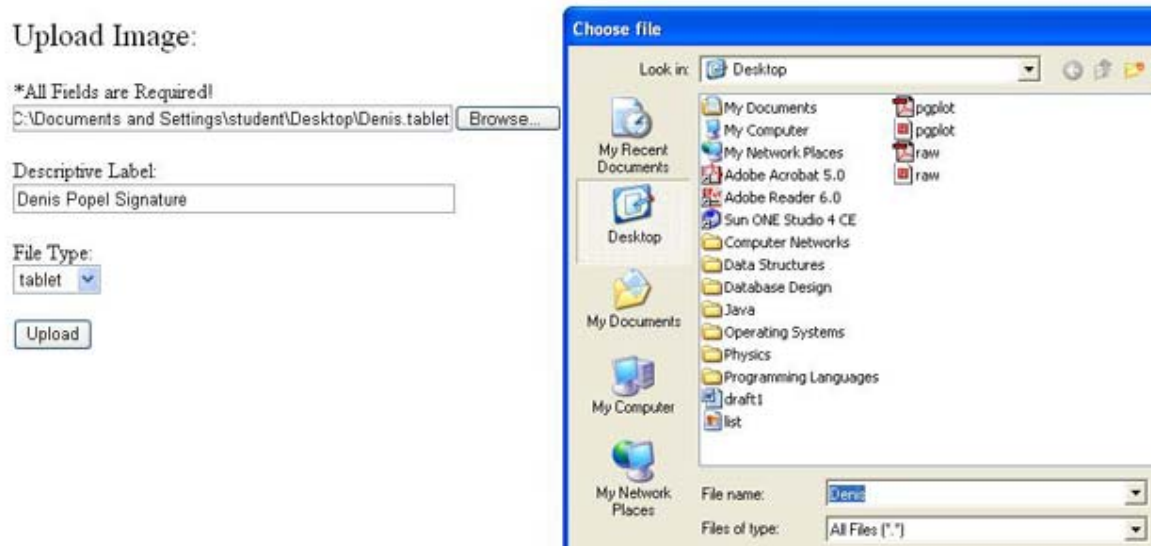


Figure 1: Upload page

into the *Images* table, generates all the necessary statistics for the image, and inserts them into the *Statistics* table.

**List script** The list script creates an HTML table, listing all the records in either the *Images* or the *Statistics* table. The interesting functionality of this list is that if the *Images* table is chosen it will read the *File* field of every record, create a thumbnail for the corresponding image, and place these data into the viewable table (see Figure 2). A button labelled “Regenerate Statistics” is also available for administrative users to automatically regenerate statistics for all images. This is especially useful when a new classification feature is added to the analysis stage. By clicking “Regenerate Statistics,” all previously entered data will be recalculated, including the new classification data, for every image in the database. There is also a delete button for all images, which allows for the deletion of an individual image and its subsequent statistics in the database.

**Add user script** The script is used to add new administrative users to the database. It requires a username, first and last name, and a duplicated password for the new user to be successfully added to the *Login* table. It also has the ability to display all the current administrative users showing the encrypted versions of their passwords (see Figure 3).

**Compare script** The only PHP function accessible to non-administrative users is the compare function, because it is the only function that does not alter the database. Compare is a simple PHP script that allows the user to upload an image, generate statistics for the image using the C++ module, and compare the generated statistics against the statistical

Select a Table:

Images 

List Table Values

Regenerate Statistics

Refresh

Table: **Images**



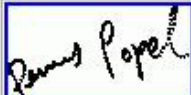
|                        | <b>Id</b> | <b>Label</b>  | <b>Extension</b> | <b>File</b>   | <b>File_type</b> | <b>Date_Added</b>   |
|------------------------|-----------|---------------|------------------|---|------------------|---------------------|
| <a href="#">Delete</a> | 22        | Denis V Popel | tablet           |  | tablet           | 2004-01-22 16:37:06 |
| <a href="#">Delete</a> | 23        | Denis V Popel | tablet           |  | tablet           | 2004-01-22 16:37:14 |
| <a href="#">Delete</a> | 24        | Denis V Popel | tablet           |  | tablet           | 2004-01-22 16:37:22 |

Figure 2: List page

data of all images in the database. The compare script then creates an HTML table displaying the testing image and listing the top ten images from the Image table and their corresponding *Similarity* ratings to the testing image (see Figure 4).

## 4 Applications

There are many business applications for which this solution would be useful. Generally, it is valuable for businesses which need multimedia databases, image databases in particular.

### 4.1 Identification

The approach outlined above is useful in image identification (including handprints, signatures, etc.) for which the system is seeking a match for an unidentified image within the image database. When identification is requested, the test image is supplied, and the database is searched record-by-record for image content statistics until a match is found. If no such match is found then a message will be returned to the user. In the event that a match is found within the database, the output would be a list of desired characteristics or other relevant information.

A good example of identification occurs in law enforcement. A fingerprint, for example,

### ADD USERS:

Username:

Password:  
(16 Character Max)

Re-Enter Password:

First Name:

Last Name:

---

### LIST USERS:

|                        | <b>Username</b> | <b>Password</b>  | <b>Last_Name</b> | <b>First_Name</b> |
|------------------------|-----------------|------------------|------------------|-------------------|
| <a href="#">Delete</a> | salexander      | 27a5149358489147 | Alexander        | Scott             |
| <a href="#">Delete</a> | dpopel          | 74ca6889592dee71 | Popel            | Denis             |
| <a href="#">Delete</a> | jstrohm         | 2c93a0da51598679 | Strohm           | Joshua            |

Figure 3: Add users page

could be captured digitally and compared against the database. The characteristics of the image will be compared against similar characteristics of other images within the database. If a match were found, then important information would be returned to the law enforcement officer, including background information of the person whose fingerprint matches the target fingerprint.

## 4.2 Verification

When verification is requested, both an image file and a label must be submitted. This label could be a person's name accompanying a signature file, or simply a descriptive keyword accompanying an image. Once the image and label are submitted for verification, the database is then searched for an appropriate match. The searching process is accelerated

## Compare Image:

### Your Signature:



### Your Results:



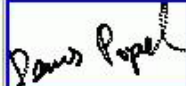
| <b>Id</b> | <b>Label</b>  | <b>Extension</b> | <b>File</b>   | <b>File_type</b> | <b>Similarity</b> |
|-----------|---------------|------------------|---|------------------|-------------------|
| 26        | Denis V Popel | tablet           |    | tablet           | 4.09083           |
| 25        | Denis V Popel | tablet           |    | tablet           | 4.39583           |
| 23        | Denis V Popel | tablet           |  | tablet           | 4.58648           |

Figure 4: Compare page

in this case because it first searches for a matching label/keyword, before it compares the image characteristics. This works faster than the identification function because it takes only data from a smaller number of image files. Since images take more time to compare than a basic text query for a label, the result is a faster query. Because the purpose here is verification, once this match is found the given output is a simple yes/no answer. Although, additional information may also be passed on to the user, it would likely be in the form of instructions for the end user (such as a request for more identification or a prompt to contact a store manager).

A good example is provided by the credit card industry. An increasing amount of commercial transactions are completed by using credit cards. This means that retailers must make an effort to ensure that the purchases are being made legally by the true credit card owner. Many large retail stores obtain the signatures required for a credit card purchase digitally. However, as of yet, there is no consistency check with the captured digital signature to ascertain authenticity. Our digital signature database solution would work well in this



situation. First, a database of authentic credit card customer signatures would need to reside on the company's server. Then, when a credit card is being used as payment, the retailer's computer would issue a request to the database server for signature authentication. Within this request, the retail computer would supply the cardholder's attributes (e.g. a credit card number) and the newly obtained digital signature to the database for comparison. The database would then compare the two signatures. If the classification returns a low correlation between the signatures, then the database server would send a message to the retailer computer stating that further identification should be requested. Another (and a similar example) is offered by the increased use of electronic documents for legal purposes. It is possible to sign electronically a legally binding document. For obvious reasons, the authenticity of this captured signature should be verified. The process required here is the verification step outlined above which will work exactly as shown for the case of the credit card company.

It is also helpful to note that although our approach was outlined to be a database identification and verification tool for signatures, other images will work as well.

## 5 Concluding remarks

This paper outlines the idea of enabling content driven searching for multimedia databases. A simple signature database has been developed with a user-friendly interface and a comprehensive comparison engine. The database is robust and easily scalable. We also listed some applications where this solution may become marketable.

## References

- [1] *MySQL*, <http://www.mysql.com>.
- [2] *PHP*, <http://www.php.net>.
- [3] L. Ullman. *PHP and MySQL for dynamic web sites*. Berkeley: Peachpit Press, 2003.

## Acknowledgements

We express our sincere appreciation to Robert Fraga (Baker University) and Alexei V Nikitin (AvaTekh LLC) for valuable comments.