UiB | Dept. of Information Science and Media Studies
UiTø | Dept. of Computer Science
NTNU | Dept. of Computer and Information Science
Telenor | Telenor R&I

CAIM
CONTEXT-AWARE IMAGE MANAGEMENT

A Metadata Editor for BergenBy – a Multi-Modal Image Database

Erik Parmann

19.10.2010

CAIM-UIB

TR #12

Contents

1 Introduction **2**
1.1 Objectives . . . 3

2 Tools used **3**

3 Design **3**

4 Features **4**
4.1 Changing between the database tables: Image_DB and Image_TMP 4
4.2 Adding Temporary Images to the DB . . . 5
4.2.1 Adding a photographer . . . 6
4.2.2 Adding keywords . . . 6
4.3 Editing Images . . . 6
4.3.1 Editing image metadata . . . 6
4.3.2 Associating images with objects . . . 7
4.4 Objects . . . 7
4.4.1 Editing an object . . . 7
4.4.2 Adding objects . . . 8

5 Bugs and the way ahead **8**
5.1 Bugs crushed . . . 8
5.2 Known bugs . . . 8
5.2.1 Username and password . . . 8
5.2.2 Input validation . . . 8
5.2.3 No atomicity . . . 9
5.2.4 Image window . . . 9
5.3 Further tasks . . . 9
5.3.1 Merging of windows . . . 9
5.3.2 Too network dependent . . . 9
5.3.3 Deleting of objects and authors . . . 9
5.3.4 Audio . . . 10

1 Introduction

This report was written to document the Metadata Editor (metadataEditor) developed for the BergenBy image database [Møller(2009)] which supports the CAIM-project (Context-Aware Image Retrieval). This second version of the Metadata was developed during the summer of 2010 by Erik Parmann and is an extension of the previous system developed by [Langøy(2008)]. Since the previous version of the Metadata editor, the DB structure has been extended to include the addition of keywords, descriptive text, audio and linking images to multiple objects. This has necessitated substantial updates to the 2008 version of the metadata editor.

1.1 Objectives

- Adding keywords. Keywords were added to the DB in 2009 and we needed the possibility of adding and editing them for both images and objects.
- Multiple objects. In 2009, functionality was added to the DB to, when applicable, link images to multiple objects. So in the new version we needed to be able to edit the objects attached to an image, both adding and removing several objects to an image.
- Objects have descriptive text attached to them, but the old version did not support adding or editing of this text. We therefore had to modify the adding and editing of objects to support the addition of text attached to the objects.
- The old version consisted of two almost identical programs, one for editing images in the temporary table and moving them to the permanent one, and one for editing of the permanent table. In the new version we wanted to merge these two clients in one program, with one common database back-end.

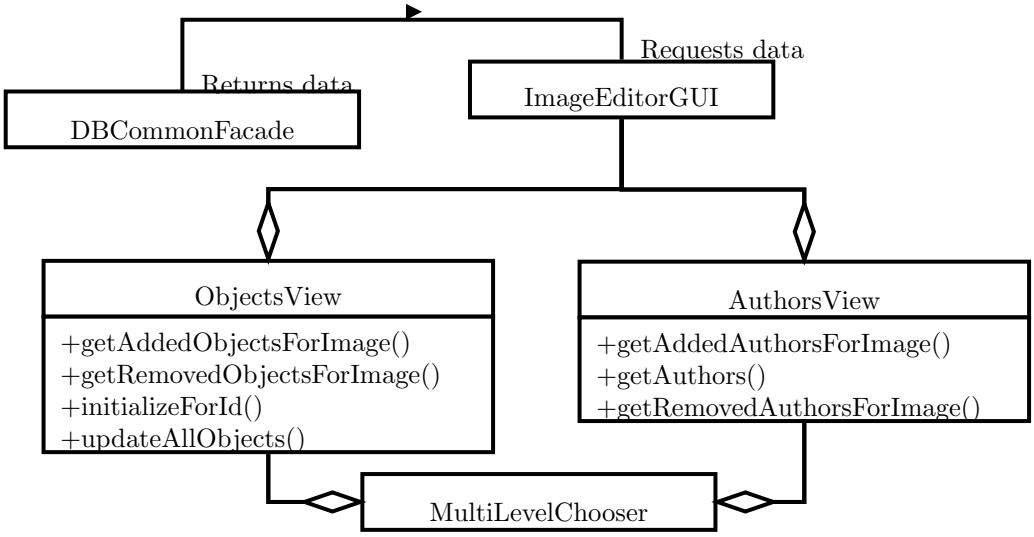
2 Tools used

Eclipse (version 3.5.0): Eclipse is a graphical integrated development environment (IDE) for Java. Eclipse supports a wide array of plug-ins, making it expandable and very customizable. For this project we only have used the subversion plugin.

Oracle SQL Developer (version 2.1.1): SQL Developer is a free, graphic tool for database development. It has been used extensively to test queries before they are added to the editor, and to get an overview of the current database-structure.

3 Design

The main class is ImageEditorGUI. This runs the main window, and handles the response from different other windows like the object editing. The communication with the database is handled by the class DBCommonFacade.java. There are also pure data classes that only carry data, like Text and Fullobject.



The class MultiLevelChooser is the back-end for the GUI that allows the user to select and deselect several object to a picture. It takes two sets of objects, one being a subset of the other. It then presents the big set as a list of check boxes, with the subset being the check boxes already ticked. Other classes can then query it for information about added or removed objects. This is also used to select the authors of text in the object-edit window, as a Text can have several authors.

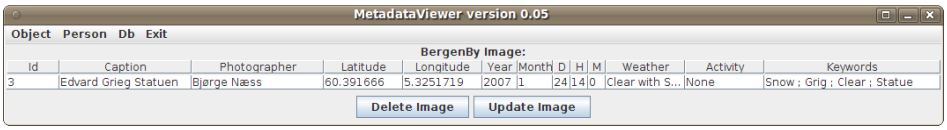
4 Features

Here we will describe the main features of the software, and how to use them.

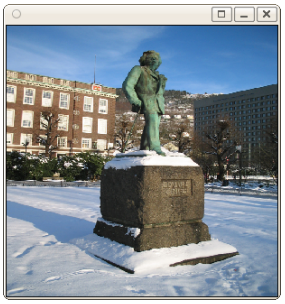
When the editor starts up the user is presented with three windows, like in Figure 1. the main window is the center of the editing, and besides changes of affiliated objects, all information about the image is edited here. The image window shows the current image that is being edited, while the object window shows the objects associated with the current image.

4.1 Changing between the database tables: Image_DB and Image_TMP

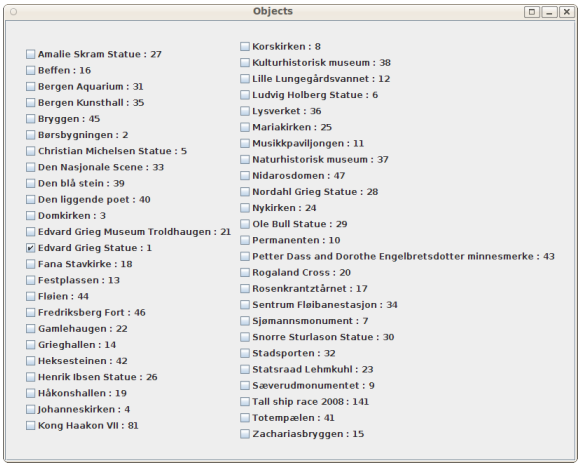
The BergenBy database contains 2 image tables. Image_tmp contains all MMIR stored images, the seed images from MMIR searches, as well as images uploaded via VISI [Døskeland(2010), Carlson(2009)]. The IMAGE_DB table contains the annotated images with such descriptive data as the date/time and location when the image was taken, keywords, the photographer, as well as links to the objects contained in the image.



(a) Main window



(b) Image window



(c) Object window

Figure 1: The three initial windows of the program

In the main window there is a menu called “Db” with two items in it, “Image_DB” and “Image_TMP”. This changes between the two database tables. When the program starts up it starts with “Image_DB”, but the user can select the Image_TMP table as necessary. In this case the “Update” button, shown in Figure 1a, is replaced with an “Insert” button, which on key-press moves the image from the temporary image table into the permanent one.

4.2 Adding Temporary Images to the DB

First select the IMAGE_TMP table from the DB menu and then the ID of the image to be annotated. Clicking on an image number will cause the image window to change to that image, and the data for that image to appear in the main editing window 1a and a clear list of current objects to appear.

Add metadata values for the Caption, weather and activity by clicking on the value field. If the image has been taken using MMIR4 [Parmann(2010)] values for Lattitude, Longitued, year, month data, hour minute have been added by MMIR4.

4.2.1 Adding a photographer

Clicking on the photographer will list current photographers that can be selected simply by clicking on their name. It is also possible to add a new photographer by simply typing in his/her name.

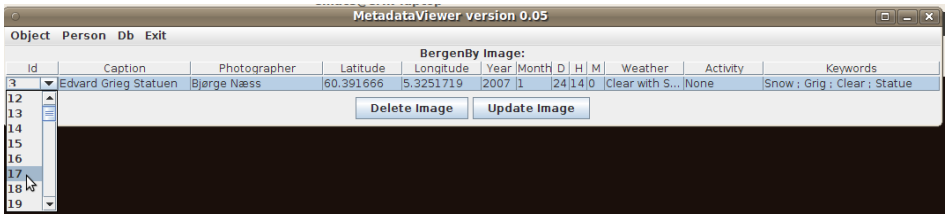
4.2.2 Adding keywords

Keywords may be single terms of multiple term phrases. Multiple keywords should be separated by a semicolon " ; ", as illustrated in 1a. If an entered word is a new keyword for the system, the user will be asked for a confirmation, and on a positive reply it will be added to the database. Afterwards press the update button to change the keywords.

When all metadata have been entered, press the "Insert" button.

4.3 Editing Images

Figure 2: Image selection



To select an image for editing, first select the IMAGE_DB from the DB menu then select the ID of the image in the main widow as shown in figure 2. Afterwards the information in the main window, image window and object window will change to accommodate the new selection.

4.3.1 Editing image metadata

To change the values of the metadata for Caption, Lattitude, Longitued, year, month data, hour minute weather and activity, simply click on the current filed value and then enter the required modification.

When editing keywords, the user is presented with a text-field with the current keywords. If there are many of them they are separated by ";", e.g

```
Statue ; Museum
```

The user can then edit this text-field, adding or removing keywords. So if the user then enters

```
Statue ; Black-white
```

the metadata editor will remove the keyword Museum and add the keyword Black-white. As noted above, the user will be asked to confirm new keywords.

4.3.2 Associating images with objects

Figure 3: Object selection

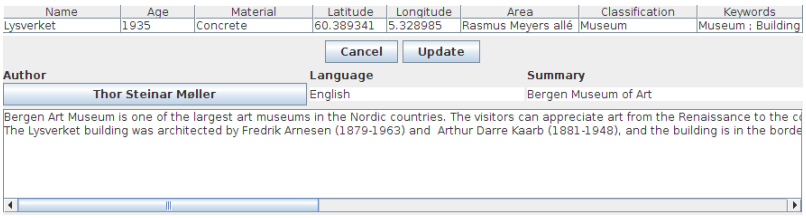
An image can have a number of objects associated with it, and which objects that are associated with a given image are shown in the object window. Here the user can select multiple objects for each image. An example of this is shown in figure 3. In this example the image contains both “Lille Lungegårdsvannet” and “Lysverket”. The user can select and deselect objects. If the user adds a new object with the add-object menu it will get added to this window. When the user presses the “Update image” the image in the database will be updated.

4.4 Objects

4.4.1 Editing an object

When choosing to edit an object, the user is first presented with a window asking the user which object to edit. See figure 4 for a screen-shot. Here the user can edit information before updating the object. There is a very low level of input-validation. Some empty fields get set to some default value (e.g if latitude is empty it is set to 0), but there is no validation. Editing object keywords is done in the same way as described in sec. 4.3.1 above.

Figure 4: Object editing



4.4.2 Adding objects

When adding a new object, the user is presented an empty object window which is identical to the one in figure 4, except that all the fields are empty. Insertion of metadata is done in the same way as for adding new image metadata or editing an existing object.

5 Bugs and the way ahead

5.1 Bugs crushed

In the report for the previous version [Langøy(2008)] describes some bugs, which we are not able to reproduce in the final version. Some have intentionally been fixed, while others have been fixed as a side-product of other changes. The bugs in question are “Comboxox”, “JTable” and “Out of Memory”.

5.2 Known bugs

5.2.1 Username and password

The username and the password for the database are stored in the source code of metadataEditor, and are extractable from the runnable jar-file. This means that anyone with access to the metadataEditor can modify the database as they want. Maybe the way to handle this is to query the user for the credentials on startup, and maybe have a way to store them locally afterwards so the user doesn’t have to write them every time.

5.2.2 Input validation

There is little to no validation of input. If values are empty we set them to default values, but beside this there is no validation. Where there are numbers to be entered we faithfully try to parse them as numbers, and crash if that fail.

The database back-end has some validation of e.g dates, and we forward errors reported by the database to the user.

5.2.3 No atomicity

The updating of an image consists of several sql-statements, one for each of the properties of the image. If there is some error in one of them (e.g the day of the month is 40) the updating of the image will stop, but the earlier updates will not be rolled back.

5.2.4 Image window

The current image is shown in a separate window. If this window is closed there is no way to get it back up except restarting the application.

5.3 Further tasks

5.3.1 Merging of windows

The current image is shown in a separate window. This was easiest given the previous design of the program, but this is a very non-standard way to handle several GUI components. It would be natural to have the image, image information and objects in one frame.

5.3.2 Too network dependent

The program queries for all it's data from the database, and on slow networks it can be a pain to use. While the nature of the program makes network traffic a necessity, things can be made better. The biggest gain can probably be made from intelligent local caching of data. Another possibility is to merge some of the queries into fewer. As an example, when the user selects an image it now does at-least 11 queries in sequence, and many of them could be merged into a single query, probably lowering the query time dramatically (as it then did not need to establish a new connection for each of them).

5.3.3 Deleting of objects and authors

It is not possible to remove either objects or authors. This since it is not always clear what to do with other things in the database connected to them. That could be Text contained in the object or that have the author as an author, or images that have that object.

5.3.4 Audio

There is no support for audio in the current version of the metadataEditor, even though objects can have audio attached to them. In a future version it could be useful to be able to listen to the records and make new ones.

References

[Carlson(2009)] C. Carlson. Visi3 - context aware image retrieval. Technical Report 8, Dept. of Information and Media Sciences, Univ. of Bergen, Sept. 2009. Available at `http://caim.uib.no/publications.shtml`.

[Døskeland(2010)] Ø. Døskeland. Visi4 - supporting relevance feedback in context aware image retrieval. Technical Report 10, Dept. of Information and Media Sciences, Univ. of Bergen, Oct. 2010. Available at `http://caim.uib.no/publications.shtml`.

[Langøy(2008)] M. Langøy. Bergenby database & metadata editor. Technical Report 5, Dept. of Information and Media Sciences, Univ. of Bergen, Aug. 2008. Available at `http://caim.uib.no/publications.shtml`.

[Møller(2009)] T. Møller. Bergen by - a multi-modal image database. Technical Report 9, Dept. of Information and Media Sciences, Univ. of Bergen, Sept. 2009. Available at `http://caim.uib.no/publications.shtml`.

[Parmann(2010)] E. Parmann. Mmir4 – mobile multimodal image retrieval (ver.4). Technical Report 11, Dept. of Information and Media Sciences, Univ. of Bergen, Oct. 2010. Available at `http://caim.uib.no/publications.shtml`.