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

MMIR4 – Mobile Multimodal Image Retrieval

Erik Parmann

October 2010

CAIM-UIB

TR #11

Contents

1 Introduction

This report was written to document the MMIR4 (Mobile Multimodal Information Retrieval 4) prototype developed for the CAIM-project (Context-Aware Image Retrieval) during the summer of 2010 by Erik Parmann. MMIR4 is the fourth version of the mobile prototype developed for the CAIM project, and is based on the MMIR3 prototype developed by Hellevang the summer of 2009[Hellevang(2008)],[Hellevang(20009)].

1.1 Objectives

The project objectives for the MMIR-4 prototype were to extend MMIR-3 and make it ready for presentation. The main changes include adding of stored search, optimizing load time and adding support for several objects per image.

The primary usage scenario for MMIR is that a user submits a picture taken with a mobile phone camera and tagged with the gps location of the user that is to be used to perform a search for explanatory information. He can also browse for a picture on the phone and use that as a seed image, but then GPS information would not be included (as we have no way of knowing where the user was when he took the picture).

Stored searches. In MMIR4 we also give the user the possibility of doing a search, but storing the result for later viewing. By default MMIR4 fetches the information needed for 4 sets of 6 pictures. This feature can also support presentation of the project outside Bergen, where it is not possible to take pictures of things in the current database.

Optimizing processing time. MMIR3 was plagued with quite long load times. On average it took 1 min 30 sec from the user chose to do a search until she received the first results and half a minute if she wanted to browse to the next 4 pictures. We also wanted to increase the amount of pictures presented in each batch from 4 to 6, and a further increase of the load time with another 50% would not be acceptable. A goal in MMIR4 was therefore to optimize load time, and make it usable with 6 pictures instead of 4.

Images with multiple objects. MMIR3 was developed when each image in the database had one and only one object associated to it. This has changed, and that change must be reflected in MMIR4. Currently MMIR3 can present text and audio for an image, but it associates only one object with each picture. In MMIR4 we want to present the user with all the objects in a picture, and let the user choose which object she wants information about.

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 use the subversion plugin, WTP and EclipseME. The last is a plugin that allows us to run MMIR4 on an emulator directly from eclipse. WTP (Web Tools Platform) is a plugin that among other things allows us to launch web applications directly from eclipse. We used this to test the middleware between the phone and the database.

Sun Java Wireless Toolkit (WTK 2.5.2) for CLDC (formerly known as J2ME Wireless Toolkit) is a toolbox for developing wireless applications that are based on J2ME's Connected Limited Device Configuration (CLDC) and Mobile Information Device Profile (MIDP). The toolkit includes the emulation environments, performance optimization and tuning features, documentation and examples.

Oracle SQL Developer (version 2.1.1): SQL Developer is a free, graphical tool for database development.

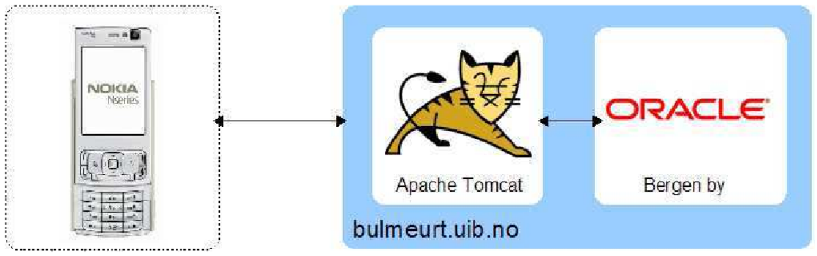
3 Design

3.1 Overview

This is the fourth prototype for image and information retrieval using a mobile phone developed at the University of Bergen for the CAIM-project. MMIR4 is based on MMIR3 [Hellevang(20009)], and uses the same architecture as the previous version. The overall design of the prototype is composed of three parts:

1. The MMIR4 client, running on a Nokia 5800 XpressMusic mobile phone.
2. The MMIR4 server application. The server runs as a servlet on bulmeurt.uib.no, inside an Apache Tomcat container.
3. The BergenBy database server, also located at bulmeurt.uib.no (developed using Oracle l0g).

Figure 1: The MMIR4 prototype. Picture originally developed by Roe Fyllingsnes and Christian Hartvedt for the MMIR prototype



From a user's perspective, the user – to – MMIR interaction is as follows:

1. He/she takes an image on his/her mobile phone and submits it as a search (also called a seed) image to the MMIR server.
2. The MMIR server returns a set of images deemed relevant by the underlying Oracle DB server. The MMIR client presents the result set as a set of 6 or 4 images per page, depending on the screen size of the phone.
3. The user can then browse through the result set and select an image containing the object that he/she wants more information about.
4. The MMIR client then sends a new request for text or audio descriptions for the object within the selected image.
5. The user then receives a text or audio description of the requested object.

3.2 MMIR4 Client

MMIR4, like MMIR3, tries to follow the MVC-framework. However, due to constraints caused by the J2ME graphics API, MVC is only partially followed. Never the less, most of the GUI-elements are divided into a view and a controller class, with most of the logic placed in the controller. MMIR functionality includes the following, with images in appendix A.

Search
This is the primary feature of MMIR. It enables the user to perform a CBIR search against the BergenBy DB, using the mobile phone's integrated camera to provide a seed image. In MMIR4 this is extended to allow the user to store the result of the search for later viewing.

Browse for image
This works like Search (without object selection), but presents the user with a menu enabling him or her to use a stored image from the phone's disk rather than taking a photo with the camera. Browse for image returns a ranked result set with images that are similar to the image used in the query.

Fork search
This enables the user to use one of the pictures in a result set as the source for a new search. The meny entry for this can be seen in figure 3f.

Subpicture selection
When the user had taken a picture with MMIR3 it allowed the user to make a rectangle in the picture and search only with that part. This option is extended to work with pictures picked up with both the "Browse for image" and "Fork search" options.

Object information
This allowes the user to read or listen to information about objects in the picture. If the picture contains several objects the user is presented with a list of them as in figure 2 and asked which to present information about. Figure 3d shows the information about "Christian Michelsen Statuen".

Draw and search
Draw and search presents the user with a white screen, 8 colours and a simple pencil-form. This makes it possible to draw simple sketches of objects using the Nokia 5800 XpressMusic mobile phone and a stylus. The sketch can be sent to the server as a CBIR query, just like with the normal Search-function.

Store image
Storage of the photo taken by the phone's integrated camera in the Bergen By database. the The user will not be presented with a result set.

Stored search

This allows the user to browse the result set(s) of an earlier search(s). By default MMIR4 stores the first 4 pages of a result set on the user's mobile phone. These can then be retrieved again as shown in figure 3e.

In addition to the previously mentioned features the client has two more options in the menu, "Settings" and "About". Settings provides the user with the option to disable / enable GPS, disable / enable downloading of multimedia (i.e. text and audio), and options related to the GPS range. The "About" feature contains information about the MMIR4 prototype as well as a list of developers.

3.3 MMIR4 server

The MMIR4 server is a Java servlet running on bulmeurt.uib.no. The server enables the MMIR4 application running on the phone to perform actions it otherwise wouldn't be able to perform both due to the limited functionality exposed through J2ME, and because of the limited processing and storage capacity of the mobile phones it is intended to run on. The MMIR4 server uses the data delivered from the client to perform queries and inserts on the database. In addition to the database that performs GPS and CBIR queries, most of the MMIR4 logic lies in the MMIR4 server.

Basically the server receives a command bundled with necessary data, and performs some action on the BergenBy DB. The command states what should be done, e.g. performing a regular CBIR+gps search, storing an image to the database or performing a search using the image of a drawing.

In a default scenario where the user is using the "Search"-feature, the server receives an image, a time-stamp from when the photo was taken, a GPS-coordinate (if applicable), and a range. The server will then perform a CBIR search on the Bergen By DB using Oracle's proprietary image search functionality.

A change from MMIR3 is that MMIR3 downloaded the audio and text for all result set images into memory, sometimes resulting in downloading image and audio for 300+ images. MMIR4 now only downloads the unique id for the text and audio descriptions, and fetches the data when requested.

4 Implementation documentation

The technical reports for both MMIR2 and MMIR3 describe their respective implementations in detail, so we will here only describe changes from MMIR3 to MMIR4.

4.1 Laziness

To deal with the long load times on MMIR3 we have changed from eager loading of image data to lazy loading. This means that the data is not loaded from the server until it is actually needed. In MMIR3, when a user did a search the client downloaded 4 full-size images with audio and text. MMIR4 instead downloads 6 thumbnails and then presents them to the user. These thumbnails are created by the MMIR4 server. If the user selects a picture for further information, the client then downloads the needed audio and text data.

The image with associated data is represented in MMIR3 as an object of type SetImage. In MMIR4 we subclass this with a class called LazySetImage which takes care of the laziness. A representative method in this class is the method for getting the image data shown in the following figure.

Listing 1: Lazy fetching of image data

```
public byte[] getImageData() {
  if (!haveImage) {
    byte[] data =
        networker.executeGetScaledImageByID(getImageId(),
             getImageWith(), getImageHeight());
    super.setImageData(data);
    haveImage = true;
  }
  return super.getImageData();
}
```

The methods for the retrieval of audio and text are similar.

The MMIR4 server also implements similar laziness when it retrieves the result from the server. In this way, it now loads only the id of the images, fetching further information as it is requested by the phone.

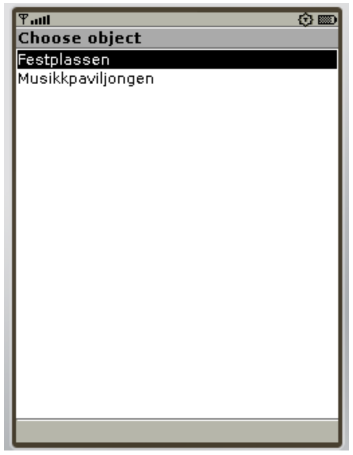
4.2 Resizing

As shown in code listing 1, there is now a method in the networker called "executeGetScaledImageByID". This queries the MMIR server for an image with a given Id and a given size. In MMIR3 the phone downloaded the large version of all the images, while MMIR4 downloads thumbnail versions of all the pictures, and only large versions when the user presses on a picture.

4.3 Multiple objects

Both the MMIR client and the server needed changes to accommodate images that can contain several objects. The data-types used to represent images now contain a list of the objects contained in that image. Both on the phone and in the server the object consists of: Id/name, Text and an audioId, while the audio itself is handled separately to save memory. In the client we needed some GUI changes as well. Now when the user chooses to view the info about an image we either show the info about the object if the image contains only one, otherwise we give a list of the objects in the image and let the user choose between them. The same holds for playing of audio.

Figure 2: The selection of an object in an image



4.4 Stored searches

After the user has taken a photo (a seed image) she can choose if she wants to do a search or a stored search. If she chooses a stored search we write the image she took to the phone, and generate an id. In addition to the first 6 pictures from the result that we present to her, we also query for the id of the 18 next pictures. These are all stored as LazySetImage objects. They are then written to the phone's memory card in a folder given by the id assigned earlier, and with names as "1.sim", where lower number means that it is earlier in the result set. As they are written as Lazy images this means that if they are pictures that have been presented to the user the image data will be stored, while if it is a picture that has not been presented to the user the image data will be fetched when needed.

In the main menu there is a new option called Stored Search, that lets the user browse the stored searches. This presents a thumbnail of all the seed images. If the user selects one of them we check the folder for that id, and fetch all the stored LazySetImage objects that are stored there. They are then presented 6 at a time, as result sets are. As they are LazySetImage objects from this point everything works as if it were a normal search, and if the user presses one of the thumbs we download data if needed.

5 Known bugs

5.1 Concurrency-bug 1 in the server

When the Caim server receives a seed-image and a request for a search it sends it to the database, where the search itself is done. Then the reply is written to a table in the database, which the server then reads. If another user searches at the same time the database might overwrite this table with the new result before the first searcher gets to read the result. Either the searching procedure should present the result directly, or it should make a new temporary row in the resulting table with an unique ID, so that two different searches get different rows, and therefore don't step on each others toes.

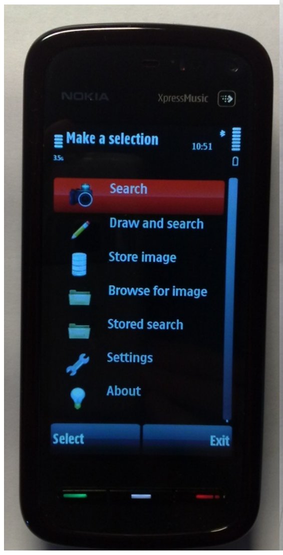
5.2 Concurrency-bug 2 in the server

After a search-image has been received by the server it queries the database for the result and stores it in an ArrayList. Then the phone sends requests like "Give me the images in image set 2", whereby the server gives those images from the List. The problem is if two phones conduct searches in the same time-span. If phone A searches first, and then phone B searches, and then phone A presses "Next" to get to the next 6 images, phone A will get the results for the search done by phone B. The way to fix this is to change the protocol of the communication between the server and the client. When the server is done searching in the database for a given seed image, instead of storing the response it should send it to the phone. This is simply a list of integers (the ID of the images), and should not present a significant increase in load-times. Then the phone queries the server for images with a given ID, not result-sets.

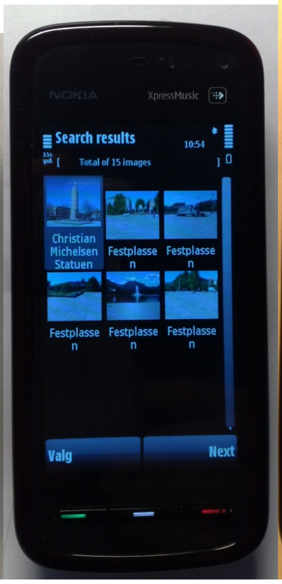
5.3 Handling of exceptions

The phone needs to interact with the file system when it stores searches and when it plays audio (as the audio is too big to keep in memory). There is no robust handling of the situation where the memory card is full. It also needs to use the phone's "Record-store" (an internal database on the phone) to store the seed images of the stored searches, and there is no handling of similar exceptional situations here. In general exceptions are caught as soon as they are thrown and rarely handled at all.

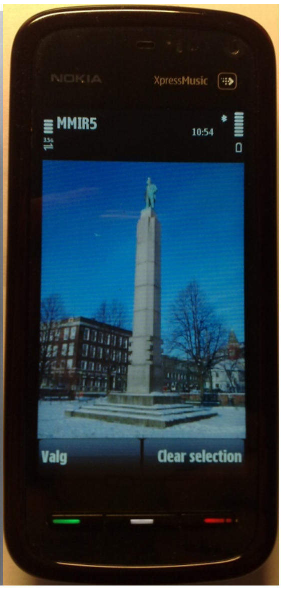
A Images



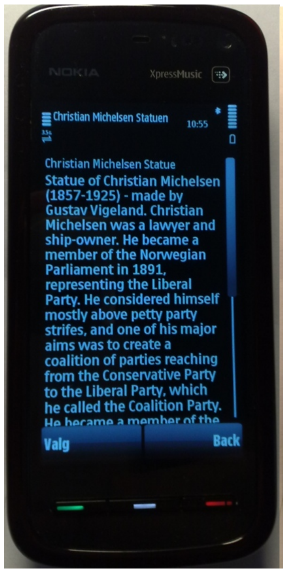
(a) Main meny



(b) Showing a result set



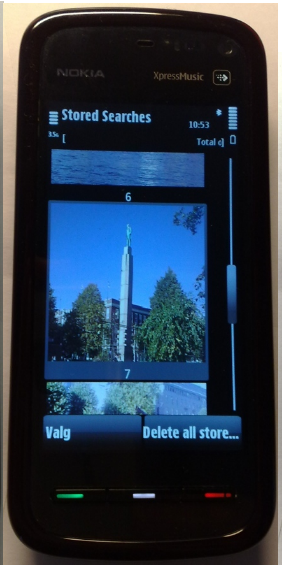
(c) Viewing an image fullscreen



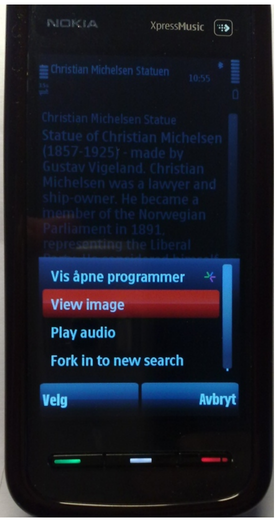
Christian Michelsen Statue

Statue of Christian Michelsen (1857-1925) - made by Gustav Vigeland. Christian Michelsen was a lawyer and ship-owner. He became a member of the Norwegian Parliament in 1891, representing the Liberal Party. He considered himself mostly above petty party strifes, and one of his major aims was to create a coalition of parties reaching from the Conservative Party to the Liberal Party, which he called the Coalition Party. He became a member of the

(d) Viewing description about an object



(e) Showing stored searches



(f) Options when viewing an result image

Figure 3: Pictures of the interface

References

[Hellevang(20009)] M. Hellevang. Mmir3 - mobile multimedia image retrieva. Technical Report 3, Dept. of Information and Media Sciences, Univ. of Bergen, Sept. 20009. Available at `http://caim.uib.no/publications.shtml`.

[Hellevang(2008)] M. Hellevang. Mmir2 - mobile multimedia image retrieva. Technical Report 2, Dept. of Information and Media Sciences, Univ. of Bergen, Sept. 2008. Available at `http://caim.uib.no/publications.shtml`.