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

VISI4

Supporting Relevance Feedback in Context Aware Image Retrieval.

Øyvind Døskeland
November 2010

CAIM-UIB

TR#10

VISI4
Supporting Relevance Feedback in Context Aware Image Retrieval.

Øyvind Døskeland
November 2010

Abstract

VISI (VORTEX Image Search Interface) is an image retrieval system that supports multiple modes of image queries: content-based (CBIR), text-based (TBIR), gps-based (GPS) and combinations of these. The result set from a VISI query is a set of images that are most similar to the query specifications according to the similarity algorithms in the underlying Oracle database management system. The user can select a relevant image from the result set and will retrieve a textual description of the object(s) shown in the image. VISI4, the 4th version of the VISI system, primarily adds support for relevance feedback, based on user selection of relevant keywords (to his/her information need) that describe his/her selected images from the initial result set. VISI4 reorders the initial result set based on the initial search image and the selected keywords, thereby improving the ***precision*** of the result set.

Development of VISI4 also included,

- Auto-generation of image keywords from the text descriptors of related objects,
- Modification of the presentation of search results,
- Implementation of user activity logs,
- Improvement of some existing features, most notably calculation of the similarity score for CBIR+TBIR queries, and
- Major refactoring of the VISI software

1 Project goals

1.1. 1 Enabling users to rearrange the order of search results.

The main goal this summer was to enable a user to manipulate the presentation order of the search result in order to increase the precision of the search result.

The image recognition algorithms provided by Oracle compare the physical features (color, texture and shape distributions) of the search image to those in the image collection. Users searching with an image with semantic information will not necessarily see this semantic information reflected in the highest ranked images in the result set. As a consequence the ordering of the result set may not reflect the desired semantic information.

The goal for VISI4 was to utilize the stored image keywords so that the user could give the system more information for ordering the result set than simply identifying relevant images. The dataflow for the goal is given in figure 1 below.

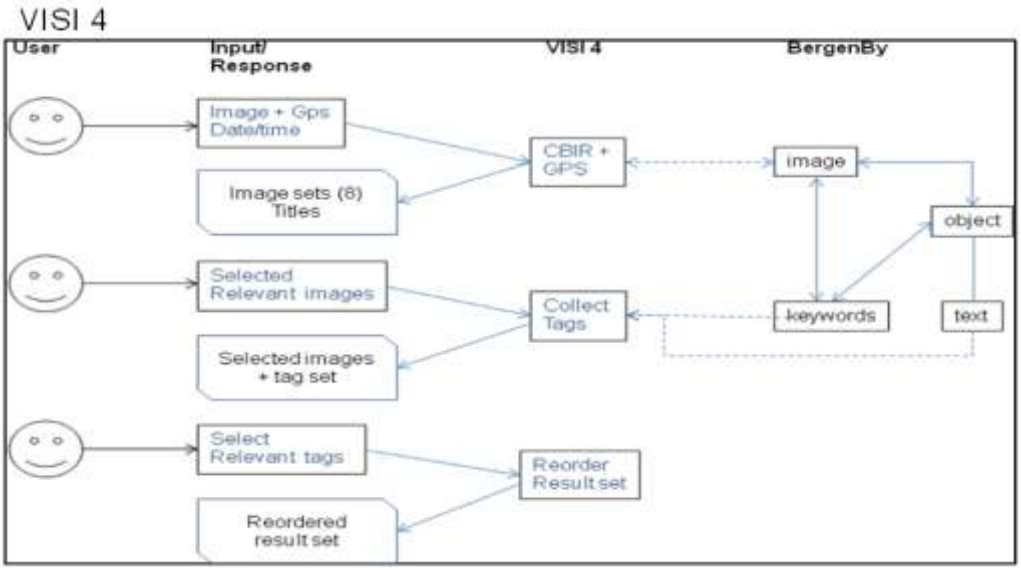

Figure1 Data flow of VISI4 reordering of result set.

1. 2 Dividing search results in groups of 8 images.

A Minor goal this summer was to change the way the images where shown in the search result. VISI3 presented a complete result set, which could be quite large. An objective of VISI4 was to enable users to determine how much of the result set that they wanted to view. A result page size of 8 images was selected. Navigation through the result set then needed to be added.

1.3 Logging usage data

A log of usage data was to be collected. This data was to include the: search image, images that the user selected as interesting, keywords the user wanted the search result to be ordered by and the ordered result set.

1.4 Improve existing features

Improve the robustness of the random image generators in case of communication error with BergenBy DB.

VISI4 should automatically store uploaded images in the TempImages Table in the BergenBy DB. VISI4 also saves the uploaded image in the tempimage folder on the Tomcat server as a backup. VISI4 is to make sure that all images are saved in at least two places to increase the robustness of the system.

Improve TBIR search to utilize all text metadata to increase recall.

1.5 Refactoring of code

Future improvements mentioned in the VISI3 rapport (Carlson 2009) were refactoring the code base. Reasons for refactoring code were primarily for improved code readability, which was necessary to understand and change the code.

2 Project documentation

2.1 Technology

2.1.1 JavaServerPages

JSP is a server side technology based on Java, which allows software developers to create dynamic web pages. JSPs can be seen of as a high-level abstraction of a servlet, and are compiled into servlets by a JSP compiler. JSPs are probably the simplest way of presenting dynamic content mixed with HTML and are better to use for presenting content than e.g. a servlet.

2.1.2 Java bean

Java Beans are reusable components. They are used to separate Business logic from the Presentation logic. Internally, a bean is just an instance of a class. A bean is used in combination with a JSP page.

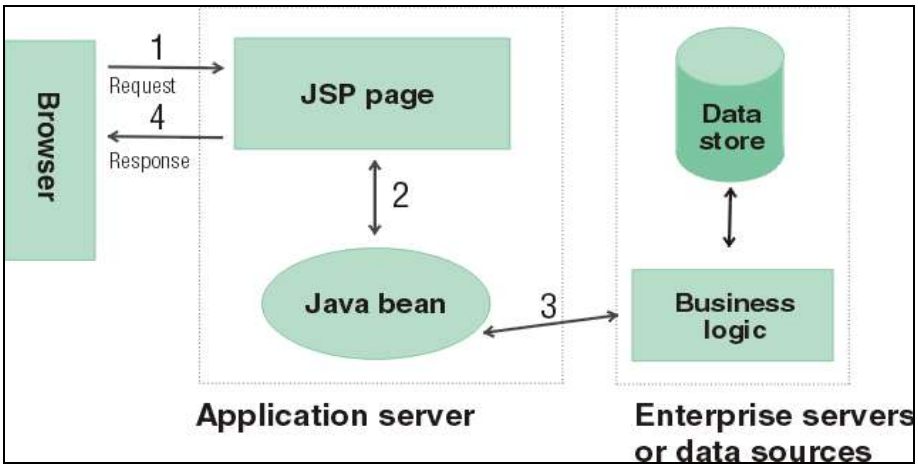

Figure 2 shows the level of JSP and Java bean in a web application.

2.1.3 Form processing

While Java Bean gives the possibility to transfer information from one page to the next page, sometimes it is more effective to transfer information by forms from one page to the next. While forms don't have the same power as a Java Bean, the integration of forms in HTML makes it much easier to make user friendly by having integrated possibilities of checkboxes, radio buttons and menus for input.

2.1.4 Tools used

- Eclipse J2EE:
- Tomcat Apache Server ver. 5.5:
- Acer Aspire
- Oracle SQL Developer
- WinSCP

Eclipse J2EE ed. was the main developer tool, this was the tool which all the Java, JSP, HTML parts were developed. The project was exported to a .war file to get deployed on the Tomcat Server 5.5.

To enable to run VISI4 as a web application, VISI was deployed on a Tomcat Server.

Acer Aspire 5600, running Windows XP, was the main computer that was used this summer.

Oracle SQL developer was used for getting information about the database, testing queries, changing procedures etc.

WinSCP is a SSH client that made it possible to access the data on the Tomcat Server.

3 Design

The code for VISI4 has been separated into three main parts: front-end (view), business logic (controller) and data model (model).

The front end is the JSP layer which gives the user a way to communicate with the system. Business logic involves the all the parts of the system that facilitate the communication between the data model and the front end. The data model manages getting and manipulating data on the Tomcat server or BergenBy DB.

3.1 Search functionality

Each of the different searches GPS (Global Positioning Search), TBIR (Text-Based Image Retrieval), CBIR (Content-Based Image Retrieval) and CBIR/TBIR (Content-Based Image Retrieval + Text-Based Image Retrieval) has its own java class, which contains the method performSearch. This method returns the set of imageIDs, represented as a java array. The tag performSearch in caimdb.tld separates between the different searches, depending on the attribute value «searchtype». The VISI3 report (Carson, 2009) goes into detail into all the different search functionalities.

3.1.1 Content-Based Image Retrieval search.

The CBIR search uses a seed image, compares the image signature to the image signatures in the database and returns a list of the images that are most alike based on their signatures. The parameters it should prioritize can be altered by adjusting the weights for texture, color, shape and spatial structure. The threshold will determine how many images are returned in the result-set.

3.1.2 Text-Based Image Retrieval Search

The TBIR search uses pure text as search parameter, and searches through the indexes for the object description (a text document that each object has) for words that are equal. In addition TBIR searches through the keywords and caption strings. The returned results are the text ids, which are linked to an object, which again are linked to one or more images.

3.1.3.1 Comparison of TBIR from VISI3 and VISI4

The TBIR search function in VISI3 was limited to only searching through the Object descriptions to find images. If an image did not have a descriptor text, it could not be found using TBIR. A descriptor could be very short and not contain the caption words, or all the keywords. Further, VISI3 used only full term matches to select relevant images.

For example: Searching for images of Domkirken using VISI3:

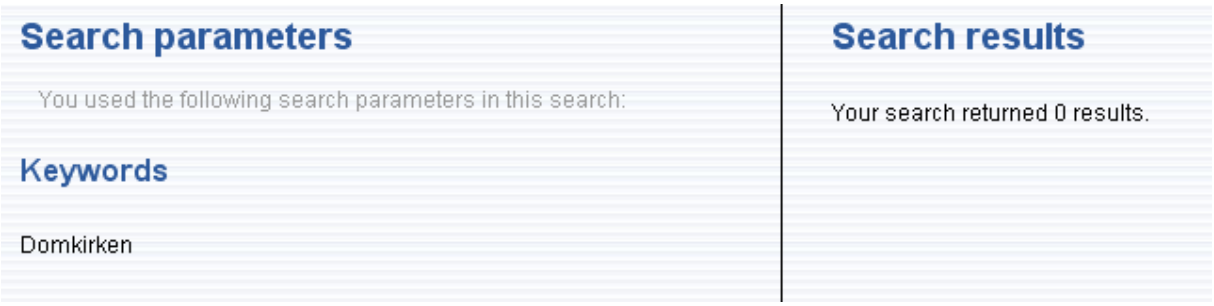

Figure 3 screenshot of VISI3 TBIR search using Domkirken as argument.

There were 0 results when using Domkirken as text argument. The reason for this was that the object description for Domkirken did not contain the term “Domkirken”, the text contained “Bergen Domkirke” or “Bergen Cathedral”. There were 14 images in the database with the caption name Domkirken. VISI4 searches all keywords and image captions and does not require whole word matches, this makes the TBIR search less vulnerable.

Example 2: Finding images of Domkirken using VISI4 and argument “Domkriken”, to compare to the former search in VISI3:

Figure 4 Screenshot of VISI4 TBIR search with "Domkriken" as argument

Figure 5 shows that in VISI4 the need for whole word match is gone. All the 14 images of Domkirken are in the result set. The search now also returns images of Nidarosdomen. The reason the search contains images of Nidarosdomen is that it contains the letters dom in order.

Example 3: Finding images of Domkirken using VISI4 and argument “dom” :

Figure 5 Screenshot of VISI4 TBIR search with "dom" as argument

3.1.3 CBIR/TBIR Search

The CBIR/TBIR function is based on the approach described in Hartvedt (2007) and consists of two separate searches, a CBIR search and a TBIR search. It then iterates through the result-sets to find the images that are in both result-sets. These images get a merged score, so that images with a high score in both of the TBIR and CBIR result sets will be the top ranking images of the CBIR/ TBIR search.

3.1.3.1 Comparison of CBIR/TBIR from VISI3 and VISI4

The CBIR/TBIR search in VISI3 relies mostly on the TBIR score when ordering the result sets. When there a weak likeness between the search string and the seed image in a CBIR/ TBIR search in VISI3 the TBIR results overpopulate the top in the result set. For example, when using an image of Sæverudmonumentet image #122, shown in Figure 6 and “Beffen” as search word in a CBIR/ TBIR search in VISI3, the results are shown in figure 7.

Figure 6 Image of Sæverudsmonumentet image #122

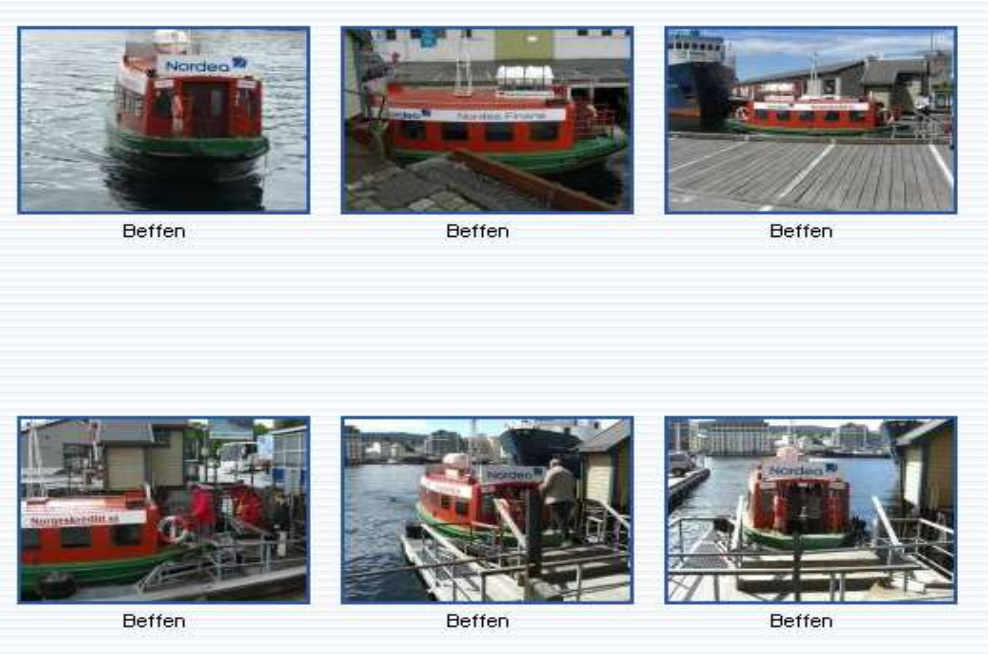

Figure 7 Screen shoot of top 6 search result of CBIR/ TBIR search VISI3

Figure 7 shows that the all the top 6 images where related to the text argument and none of the seed image. When using the same arguments in VISI4 the result as shown in figure 8 the population is mixed with images that scored high on TBIR and scored high on CBIR.

Figure 8 Screen shoot of top 6 search result of CBIR/ TBIR search VISI4

There is of course not enough data in this example to conclude anything, but it seems like VISI4 returns images that look more like the seed image. The source code for the CBIR/TBIR search function is located under the handlers.search.CBIRTBIRSearch class.

3.1.3 GPS Search

The GPS Search uses the same Oracle procedure as VISI2 and VISI3. The source code is located under the handlers.search.GPSSearch class. And discussed in the VISI 2 report (Rørvik, 2008).

4 New functionality

4.1 Reordering of a search result set

The main focus for the summer 2010 VISI project has been to add the possibility to reorder a search result set. In VISI3 the search process stops when the first result set is given, as shown below in Figure 9.

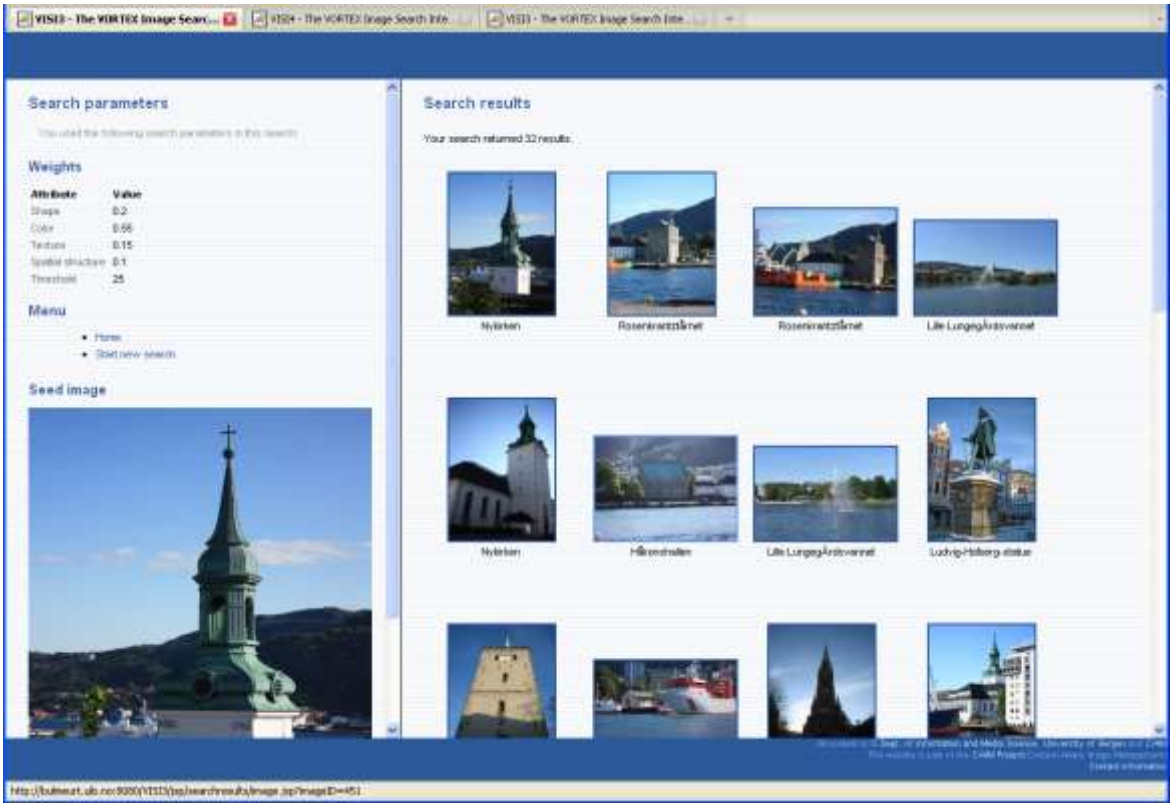

Figure 9 Screen shoot of the search process VISI3

In VISI4 the users have a possibility to select images they feel are interesting, and they can get the keywords from those images. This is illustrated in Figure 10. Each image has a checkbox next to it and a “Get keywords from selected pictures” button on the under the images.

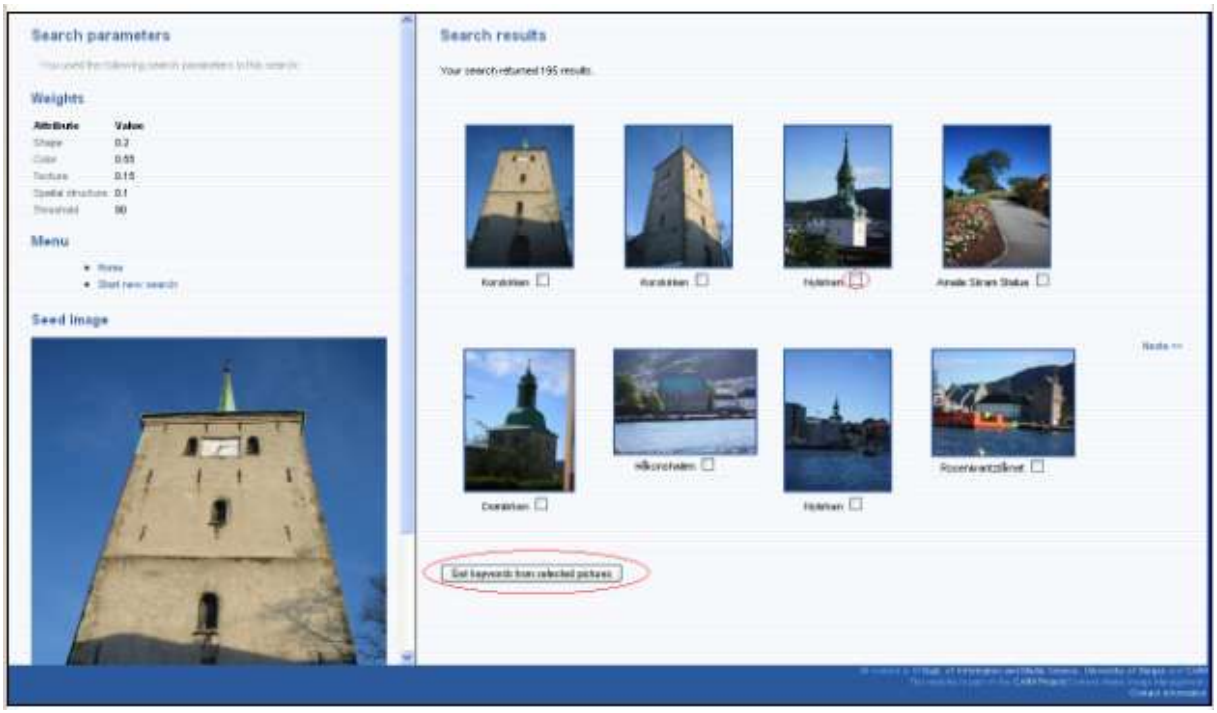

Figure 10 Screen shot of search process VISI4

Selecting images by clicking the checkbox beside the caption title, and then pressing the “Get keywords from selected picture” button brings the user to the next page in the reordering process. The user can select any number of images, all the keywords from the selected images will then be returned to the user. Figure 11 gives the user a list of the keywords for the image number 3 “Nykirken” which has a red circle surrounding the checkbox in Figure 10 when it is selected.

Figure 11 Screen shoot of keywords from images VISI4

The user is now given two lists to choose from. The "Keywords from images list" consist of the caption and keywords of the selected images. If several images are selected that have the same keywords the keywords will not be listed twice. The "Seasons and location keywords" use the date/time and location metadata of the images in BergenBy DB to order by the time of the year the images were taken, or the geo data of where the images where taken. If the user was interested in images of *bell towers*, the user could select the keyword bell tower and the former result set is reordered so that images that have the keyword bellower will come first. The reordered result, shown in Figure 12 improved the precision dramatically. The first image that is not of a bell tower is after the reordering image number 21 in the result set, and before the reordering it was number 4 in the result set.

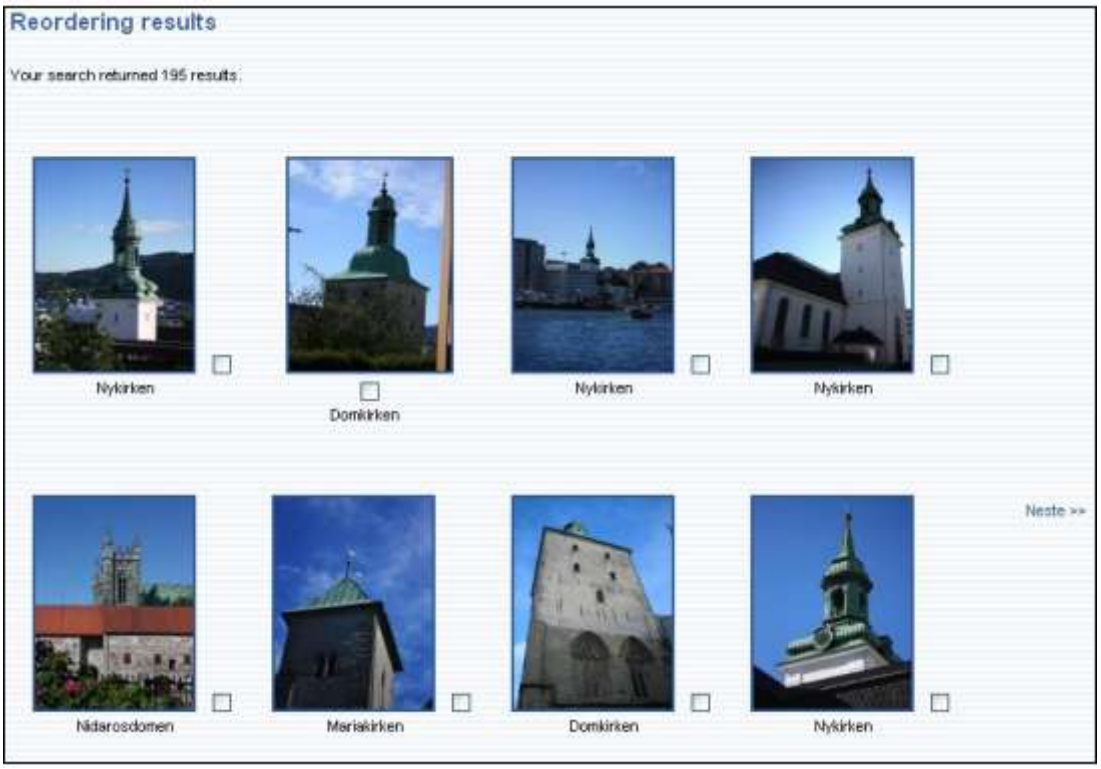

Figure 12 Screen shoot of reordering of result set VISI4

The original ordering of a CBIR search result will be used, If several images have the same number of keyword points, the image that had the best score in the original CBIR search result will be the first image in the reordered result set. Figure 13 shows the interaction between the user and VISI4 showing that it is possible to reorder a reordered result set.

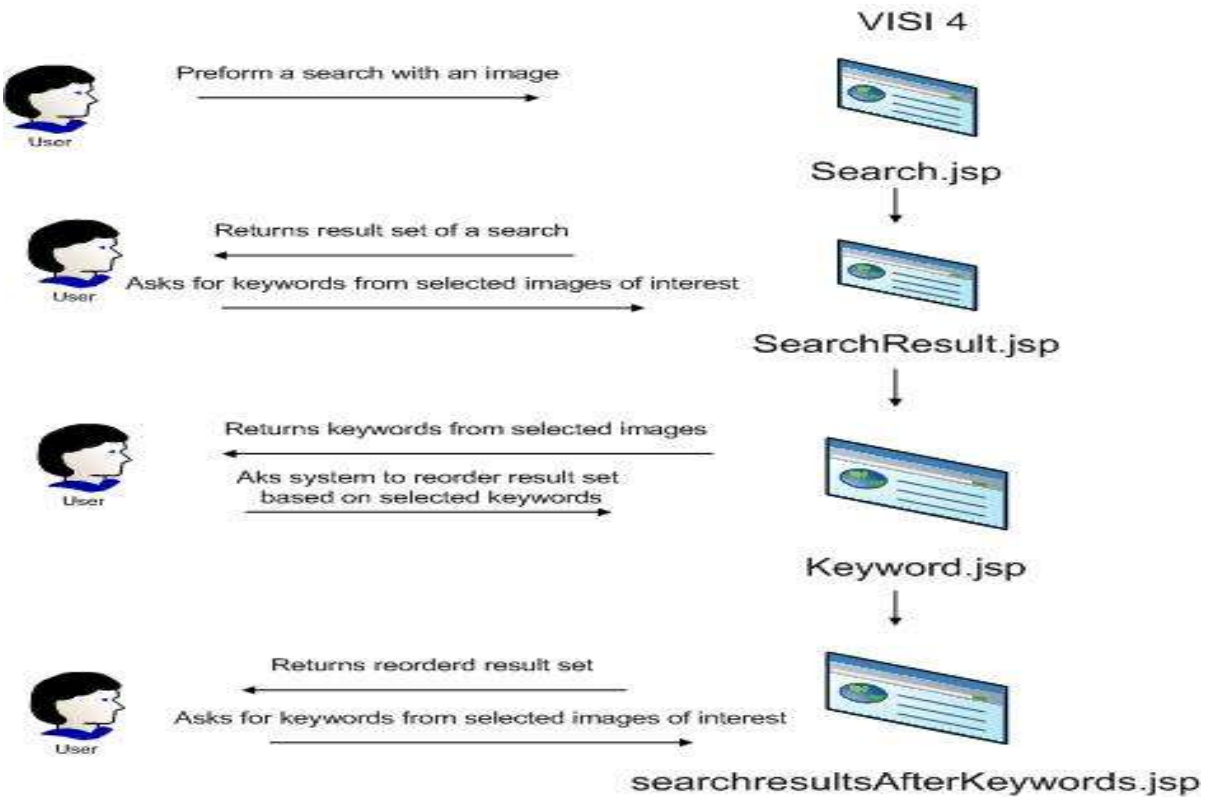

Figure 13 Dataflow between user and VISI4 in reordering a result set

4.2 User data recording

Evaluation of VISI4's ability to reorder a result set will be based on use by a number of persons. As discussed under point 1.3, the user data to be collected includes; the image argument, the images selected, the keywords selected and the reordered result set. VISI4 writes a text file that includes all the user data using the path /opt/apache-tomcat-5.5.23/Userdata. The text file's name starts with "dataFromUsers" and is followed by the date and time of day that the file is written. It is only when a user reorderes a result set that user data is recorded. It is the searchresultsAfterKeywords.jsp page that writes the user data file.

5 Future development

TBIR search has been improved in VISI4, but there are still improvements that can be made. TBIR should in the future support fuzzy search and improve weighting if the text argument was a match with the keyword and the caption.

The VISI4 user interface should be reworked. The frontend has not changed since the first version of VISI. The frontend was not designed to include more than one step in a search process; this creates some problems that should be looked at in future development. The

information panel that is located on the left changes during the search process, and completely disappears when the user gets the reordered result set.

The *upload image* feature should be able to utilize an image link, so if user finds an image they want to search with when surfing the web, they can just paste in the web URL of the image and use the linked image as a seed image.

An RSS feed could be added to a search result, so if new images are added in the BergenBy DB that matche the search criteria the user will be alerted.

To improve precision users could be able to burry an image in the result set so that the result set would for a given image improve over time.

Refactoring of code is needed to increase future development speed. The readability of the code in the VISI4 project is at a low level. In several places documentation is not correct for the current code. This makes the project hard to understand when reading code, and difficult to improve existing features.

The GPS search has an error in computing distance which should be corrected in future development.

When uploading an image, the user should be able to add GPS data from a world map, prior to using it as a seed image.

In a future version of VISI, the result set could be supplemented with new images that match the selected keywords.

Acknowledgements

The work on developing VISI4 and this report has been a team effort. Christian Hartvedt has designed the specifications for the user feedback strategy implemented in VISI4. Øyvind Døskeland has done the programming of both the new and revised features necessary to support this form for relevance feedback. Joan Nordbotten has lead the project and edited this report.

References

Carlson, Christoph (2009). “VISI 3: Context Aware Image Retrieval” http://caim.uib.no/publications/VISI3_TR.pdf

Hartvedt, Christian (2007). “Utilizing Context in Ranking Results from Distributed Image Retrieval”. *Norwegian Informatics Conference*, NIK, Nov.19-21, 2007.

Rørvik, Anders (2008) **“**VISI2 - Combining CBIR Search with Text and GPS Location Criteria.” http://caim.uib.no/publications/VISI2_report.pdf