JspMuvis - An Approach to Displaying Web Based Images on Connected Mobile Devices

Malik Yasir Amin

University of Tampere
Department of Computer Sciences
M.Sc. Thesis

Supervisor: Zheying Zhang
June 2008

University of Tampere
Department of Computer Sciences
Malik Yasir Amin: JspMuvis, an Approach to Displaying Web Based Images on Connected Mobile Devices
M.Sc. thesis, 45 pages, 7 index and appendix pages
June 2008

Recent multimedia mobile devices are equipped with camera and higher storage capabilities. With new multimedia mobile devices users can generate and consume multimedia (image/audio/video) items on the fly and hence this causes a rapid growth for the personal multimedia collections. Consequently it brings the needs and ways for the efficient management and accessibility of such personal archives. Scientists and researchers have undertaken the challenge of meeting user requirements for content management over mobile devices. Efficient and reliable systems have been developed for search and retrieval of multimedia content. However, the area of generation of specific content for individual devices needs more attention.

Different mobile devices have different input, output, hardware, software, and network capabilities. JavaScript support is not commonly available, so is the support for Java. Display of content on the small screens and with limited capability browsers is still not mature enough. Opera in Small-Screen mode will not display many images and resize others to fit the screen. Dynamic selection of Image and Video has to be generated according to the size and capabilities of the device. Also the content provider must have one interface which is capable to adopt according to the capabilities of the device. Large images are not useful on mobile devices, and those larger than screen will make for a worse user experience. The images shouldn't be larger than about half screen size (about 150-100 pixels). The display of content on different devices with different screen sizes is a challenge. A solution with common interface is required that will generate the content according to the capabilities of mobile devices.

This thesis presents JspMuvis framework, that is capable of detecting the capabilities of mobile device and generates customized multimedia content. The proposed framework is a client-server architecture, where server is active on Personal Computer (PC) and an Internet browser application acts as client on the mobile devices. JspMuvis keeps the client side processing to minimum level, that is only displaying the content, and this makes JspMuvis browser dependent but not phone dependent. Content generated by the web server will be displayed differently in different phone browsers, depending upon their capabilities. In order for a web server or web-based application to

provide optimized content to different clients it requires a description of the client capabilities.

The solution provided by JspMuvis resolves the Java dependency on mobile devices as well. Since all the processing is being done on server side, which is a powerful computer, the only functionality to be performed on mobile devices should be display of the content.

Key words and terms: M.Sc. thesis, Mobile Devices, Images, videos, JspMuvis

Contents

1. Introduction

Usage of mobile devices is increasing every day and so is the processing power of these devices. Capabilities of mobile phones, smart phones, and personal digital assistants are increasing along with the new emerging network technologies. A few years ago, the only way to access the Internet was through a personal computer or workstation. Since the middle of year 2000, the number of different kinds of device that can access the Internet has grown from a small number with essentially the same core capabilities to many hundreds with a wide variety of different capabilities. However, there are variations between the facilities offered by various internet browsers in those devices, some being capable of use on text-based terminals and some of reasonably large color display with full graphic capabilities. Presently the network operators are providing larger and cheaper bandwidth. And the internet usage has increased with internet browser bearing mobile devices. This evolves the need of device independent internet content.

Modern multimedia (image/audio/video) mobile devices with integrated support of camera provide new multimedia services, which are becoming an essential part of our daily life. Figure 1 shows some of the mobile devices that are equipped with latest technologies. 3G [1] networks are already in the market and offers new services. With new multimedia mobile devices users can generate and consume multimedia items on the fly and hence this causes a rapid growth for the personal multimedia collections. Consequently it brings the needs and ways for the efficient management and accessibility of such personal archives.

Usage of mobile device for web browsing has become a daily life need. Although the capabilities of mobile devices are increasing everyday still the presentation multimedia content in web browsers has not reached maturity. Most of the times the content doesn't appear well formed due to different screen sizes. For example, reading news on mobile device screen becomes extremely difficult when scrolling in different directions is needed due to different sized images present in the web page. Some of the websites have launched their version of websites which are only for mobile device users, launching a different site with same content would not be the best solution to resolve the problem. The web content providers need a way to provide presentable content which is suitable for personal computers and the mobile devices at the same time.

Figure 1: Examples of different multimedia mobile devices

Significant amount of work has been done in the field of content based multimedia search. Robust systems are being developed to search the web or other databases for matching images, videos and audio content. However, making the search results presentable on mobile devices still remains as a challenge. This work concentrates on finding the answers to following question

- How can we display web based images and videos on any connected mobile device's screen in an optimal way?

In the following sections, we shall discuss some of the available technologies and the existing work in the field of displaying web based images and videos on connected mobile devices. We shall evaluate some of the existing solutions and see how the display of images and videos on mobile devices is possible in an optimal way. The capabilities of mobile device e.g. processing power, memory and battery limitations play major role in the field of content based image and video retrieval. We keep the limitations of mobile devices in mind while looking for the answers of the question mentioned above. Since the content based search and retrieval requires processing power and memory, a good solution might be the one that does not use mobile device for content based processing. We shall also propose an implemented solution which can be used to retrieve and display the image and video content on mobile devices. Again keeping in mind the limitations of mobile devices, the resource consuming operations should be done before final content reaches the mobile device. At the end we shall evaluate if the existing technologies, scientific work and the proposed solution can provide the answers to above research question.

2. Approaches to Displaying web based Images on Mobile Devices

Web content is the textual, visual or aural content that is encountered as part of the user experience on websites. It may include, among other things: text, images, sounds, videos and animations. Multimedia content includes a combination of text, audio, still images, animation, video, and interactivity content forms.

Capabilities of mobile devices include processing power, amount of physical memory, screen size, screen resolution, color support, installed operating system, support of different technologies e.g. Java, network support and battery time etc. Some of the recent mobile devices are running Microsoft Windows CE [2] operating system equipped with Microsoft internet browser others could have Symbian [3] as operating system and Opera mini browser [4]. With Microsoft internet browser users can visit the Web sites they as they browse them from their desktop computers, using Web address formats they are already accustomed to. Opera Mini is especially designed for mobile devices and is able to render real Web pages and available for nearly every mobile phone in use today. However, the functionality to render and adjust the images according to mobile device capabilities is not present in these browsers.

Some mobile devices have small displays with limited graphics capabilities, while other phones and WAP-enabled Personal Digital Assistants (PDAs) have Quarter Video Graphics Array (QVGA) (320x240) screens and up to 16 Million colors. A Web-enabled mobile phone running a Wireless Access Protocol (WAP) browser is expecting a form of the Wireless Markup Language (WML). Even though WML is an Extensible Markup Language (XML) document type, there are still significant differences in how individual internet browsers render WML on different devices.

With all the existing different types of technologies and software, it is very hard to follow one standard that can be used to display content. Every device would require customized formatting of images (resizing etc.) before these can be displayed in a proper way. If news content is prepared in a way that it is displayable on a certain number of mobile devices, it might appear in a completely unreadable form on some other new coming devices. So due to different capabilities of mobile devices, displaying of images in an optimal way is a challenge.

Keeping the existing work in mind the displaying of images with customized formatting can be handled with more than one approach. We shall discuss some of these approaches and evaluate which one can be the optimal choice.

1. Image Formatting on Mobile Devices
2. Using Scripting Languages and XML schemas to display images
3. Formatting images on powerful computers before they reach Mobile Devices

2.1. Image Formatting on Mobile Devices

Sun micro systems [5] provides libraries and extensions that can be used by software applications to retrieve the capability information (operating system, screen resolution, screen size, color information etc.) of the mobile devices. A software application can be installed on the mobile device that can process and format any image before it is displayed to the screen. This application can extract the capabilities of the mobile device on the run time or it can store the capability data at the time of its' installation. A similar solution, using Java programming language [5], has already been developed at Tampere University of Technology.

No doubt that with this approach images can be displayed in proper format and size specific to any mobile device. However, this approach demands the processor and memory resources. Furthermore, it consumes large amount of battery which reduces the talk and standby time of the device. And since a common user is not used to install software applications to mobile devices, this approach does not appear to be very practical.

2.2. Using Scripting Languages and XML schemas to display images

Most of the times internet content is accessed via internet browser application, which is present inside the mobile device. The internet browser displays the content using Hypertext Markup Language (HTML [6]). HTML can also include instructions to format the internet content, e.g. color, bold, size etc. These formatting instructions can be written separately and attached to any web site page as style sheets. An obvious solution would be to format the content according to a specific XML [7] schema and use XML style sheets to adapt the content to a specific device. An intermediate XML format can be used that represents the full structured data. This intermediate format can be generated from any number of sources, an intermediate application then itself can serve as the source of possibly multiple output formats, translating first to the intermediate XML format, then finally to a display format (HTML). For example W3C [8] recommends an XML [7] based language, Scalable Vector Graphics (SVG) [9], that describes interactive vector graphics, text, images, animation and graphical applications in XML. Appendix B shows another example of using XML schema to display custom images on mobile devices.

However, this solution would require style sheets for every combination of browser/markup/device, which can number in hundreds. Also the process can be extremely low if the intermediate processing of XML format is done on mobile device.

2.3. Format images on powerful computers before they reach Mobile Devices

Whenever a mobile device connects to internet and makes request for web content, and web browser sends an HTTP request [10] to the particular website, it attaches certain set of information to the request, called User Agent [11]. User agent is a part of the HTTP request as a string, prefixed with User-agent: or User-Agent. The information present in a typical User Agent string is called User Agent Profile [12]. It typically includes device capability data such as, internet browser name, internet browser version, network capabilities, hosting operating system, language, screen size etc. This information can be used by web site to form the web content in a way that it is presentable on the mobile device screen.

The format of User Agent Profile is a standard defined by OMA [13]. The web server can analyze the user agent profile before it responds with the web content. The images in web content can be processed according to device capabilities on web server. And since the web servers are normally running on powerful computers, the image processing can be very fast.

Using the device information from User-Agent to format text based web content for mobile device screens is fairly simple, however the multimedia content would require extra processing which makes the multimedia content, for example images, to fit into the small and different sizes of screens of mobile phones. Considering the limited processing power of the mobile devices multimedia content cannot be processed on mobile device. The resizing of the images can be done on powerful computers and their size is reduced to fit the smaller screens of mobile devices before they are delivered to a particular mobile device. This would reduce the amount of data being transferred over the network so the communication becomes faster.

This approach does not require any processing on mobile device, hence not using any of its resources. Images, when they are received on a mobile device, along with other web content, are already in a shape that fits with rest of content. Figure 2 shows the communication between a mobile device and a typical web server.

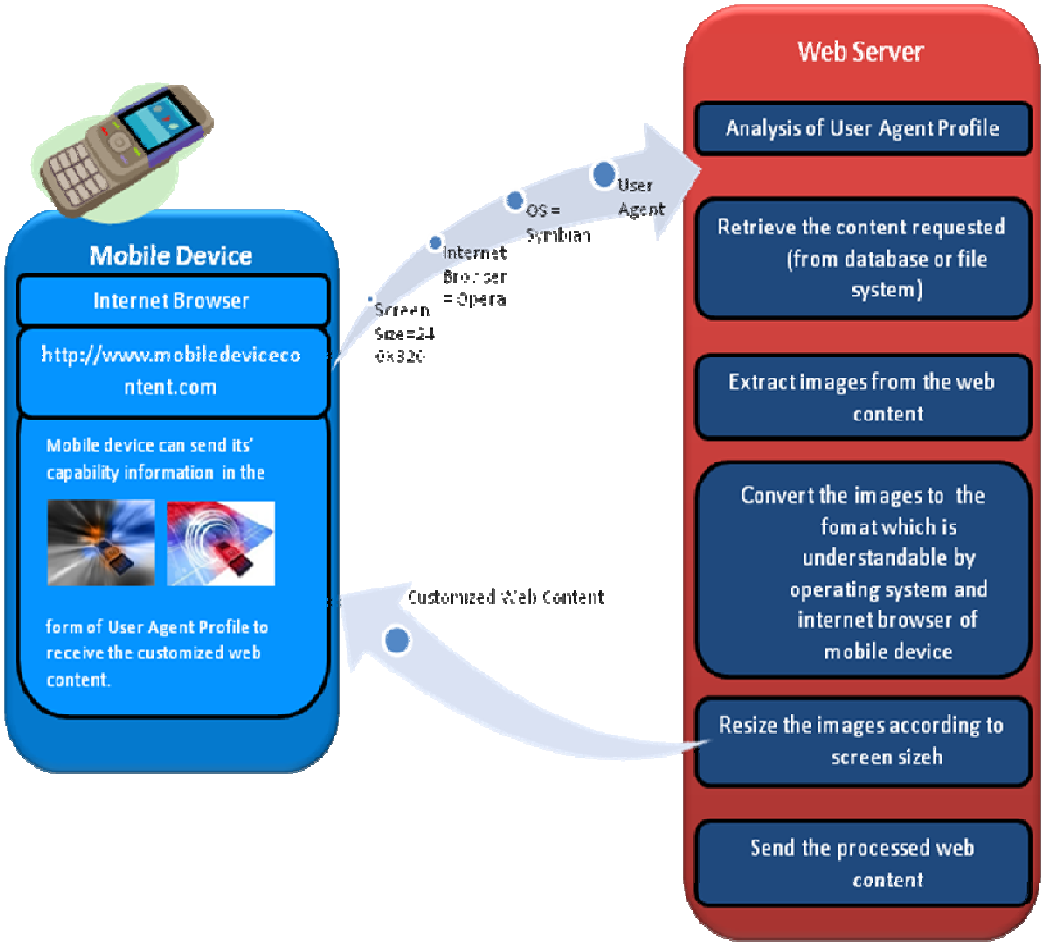

Figure 2: Formatting images on Web Server (running on powerful computers) before they reach Mobile Devices

Displaying images on mobile devices is a challenge due to their different formats and sizes and due to hundreds of different kinds of devices with different capabilities. The limited resources of mobile devices is also a major factor when it performs do image processing. The discussion in this chapter reveals that images need to be formatted to fit to smaller screens and the best approach could be to format images on web servers (running on powerful computers) according to available device capability information.

3. Existing Work

The generation and dissemination of digital multimedia content is gaining a phenomenal growth as the digital world advances. The storage technology and the advent of the World Wide Web has been the causes of increased amount and complexity of digital information being generated, analyzed, stored, accessed and transmitted. However, this rate of growth has not been matched by the simultaneous emergence of technologies that can manage the content efficiently. State of the art systems for multimedia content display on mobile devices lag far behind the expectations of the human users of such systems. The users mainly expect these systems to perform analysis at the same level of complexity and semantics that a human would perceive while analyzing the content.

A significant amount of work has been done to define the common standards which can be used by any mobile device to express its' capabilities. World Wide Web Consortium (W3C) [8] created a model to represent mobile device capabilities called Composite Capabilities / Preferences Profile (CC/PP) [14]. Open Mobile Alliance (OMA) [13] developed User Agent Profile (UAProf) [12] which can be used by any internet browser to provide mobile device capability information. 'Systems, Applications and Products in Data Processing' (SAP) [15] provides libraries for mobile extensions to the Java Servlet Container. These libraries allow the development of device specific web applications for mobile devices in the Java programming language. By using these extensions, the attributes and capabilities of the mobile device making the request can be obtained, and then taken into account when the Web application is displayed. Moreover, the extensions support the use of Web controls when developing Web applications that can run on mobile devices, allowing you to program Web applications that are browser and device-independent. Google offers email client application for mobile phones [16] and British Broadcasting Company (BBC) has mobile version of news content [17]. However displaying the multimedia content (images, videos) in a customized way needs more maturity.

Content-Based Image and Video Retrieval (CBIVR) is an active area of research for past decade due to its promising results. In CBIVR various low-level features color, texture, etc. are used for dis-similarity distance calculation. Low-level visual features (descriptors) are extracted from the images and stored in a database. Using such features, query by example (QBE) [18] based retrieval performs relatively well for images. Systems such as "Multimedia Video Indexing and Retrieval System" (MUVIS) [19], VisualSEEk [20], Photobook [22], Virage [23] have common feature of having a framework for indexing and retrieving over multimedia databases. Particularly the contemporary MUVIS is developed as a framework for content-based multimedia indexing and retrieval on a PC-based environment. MUVIS provides a unified and global

framework and consists of robust set of applications for capturing, recording, content-based indexing and retrieval, combined with browsing and various other visual and semantic capabilities.

With the help of camera in mobile device user can capture an image and perform content-based query operation virtually from anywhere. However CBIVR for mobile devices adds new challenges beside a content-based query operation. For instance different mobile devices come in different design and capabilities; moreover, they have different operating systems and input/output limitations. So it is hard to provide a generic content-based multimedia indexing and retrieval solution that suites all devices.

Recently the capabilities of mobile devices have improved significantly (faster input/output, memory capacity and processing) but comparatively they are still lacking far behind the desktop computers. With the existing mobile operating systems such as Symbian OS [3], Window CE [2], Linux, etc. it is possible to index a multimedia database entirely in a mobile device [23] but there are many limitations. For example mobile devices have propriety Application Programming Interfaces (APIs) for handling (i.e. accessing, processing, editing, streaming, etc.) multimedia items. Applications using such proprietary APIs will be limited to certain set of devices or certain platforms (operating systems). Another limitation is battery power consumption and lack of system resources such as processing power and memory capacity. Above all with the ever-increasing number of multimedia items on a mobile device, it might take an infeasible amount of time to perform content-based indexing and retrieval operations due to lack of speed and memory. Furthermore, such a system would consume significant battery power that eventually reduces the mobile device talk and standby time. Therefore, a feasible solution to this problem could be performing all the resource-taking indexing tasks on powerful machines (i.e. server in a computer) whilst the mobile devices act as their clients. M-MUVIS [24] is existing client-server architecture with Java Servlets [25] on the server side and a stand-alone Java [5] application on the client side. It consists of Servlets which are deployed over Tom-Cat [26] web server active on the PC and the Java application, so called midlet [27], which can be deployed on any Java enabled device.

3.1. Image-based Deixis for Finding Location

The multimedia item name that is automatically assigned by the capturing software on the mobile device is not so useful for media content recognition. Annotation of digital media at the time of capture is a cumbersome process for a mobile user, which is the basis of metadata creation process [28]. Image-based Deixis for Finding Location (IDeixis) [29] is content-based image retrieval for location based services. With IDeixis a mobile device user can capture an image and send to the server via Multimedia Message Service (MMS) for content based search. IDeixis server searches the website containing the similar images and responds with five most relevant images. The images are presented in small thumbnails having links to their source web pages. User can click on the thumbnail and IDeixis will take user to the website that contains the resultant image.

IDeixis produces the search results as thumbnails. However, it doesn't incorporate the device capabilities, e.g. screen size, resolution and image format support. Furthermore, users usually experience a large latency because of an inadequate mechanism for image transport used in IDeixis, which is based on MMS. Therefore, without proper content-based management methodologies applied, it is not quite feasible to find and display a particular media item whenever needed.

3.2. Multimedia Video Indexing and Retrieval System (MUVIS)

Multimedia Video Indexing and Retrieval System (MUVIS) [19] is a framework for the multimedia content management. It is developed at Tampere University of Technology. MUVIS provide several functionalities to manage multimedia content. These functionalities include indexing, browsing, querying, summarization, etc. of the multimedia collections such as audio/video clips and still images.

MUVIS provides a well-defined interface to integrate dynamically (in run time) visual/aural feature extraction (FeX/AFeX) algorithms and it hosts applications for real-time audio and video capturing, encoding, database creation, multimedia conversion, indexing and retrieval.

MUVIS framework is based upon three applications, Audio Video Database (AVDatabase), Database Editor (DbsEditor), and Muvis Browser (MBrowser). Each of these applications has different responsibilities and functionalities. AVDatabase is mainly responsible for real-time audio/video database creation with which audio/video clips are captured, (possibly) encoded and recorded in real-time from any peripheral audio and video devices connected to a computer. DbsEditor performs the indexing of the multimedia databases and therefore, offline feature extraction process over the multimedia collections is its main task. MBrowser is the primary media browser and retrieval application into which progressive query (PQ) [18] technique is integrated as the primary retrieval scheme. Normal query (NQ) [18] is the alternative query scheme

within MBrowser. Both PQ and NQ can be used for retrieval of the multimedia primitives with respect to their similarity to a queried media item (an audio/video clip, a video frame or an image). Due to their unknown duration, which might cause impractical indexing times for an online query process, in order to query an (external) audio/video clip, it should first be appended (offline operation) to a MUVIS database upon which a query can then be performed. There is no such necessity for images; any digital image (inclusive or exclusive to the active database) can be queried within the active database. The similarity distances will be calculated by the particular functions, each of which is implemented in the corresponding visual/aural feature extraction (FeX or AFeX) modules.

MUVIS databases are formed using the variety of multimedia types belonging to MUVIS multimedia family. The associated MUVIS application will allow the user to create an audio/video MUVIS database in real time via capturing or by converting into any of the specified format within MUVIS multimedia family. Since both audio and video formats are the most popular and widely used formats, a native clip with the supported format can be directly inserted into a MUVIS database without any conversion. This is also true for the images but if the conversion is required by the user anyway, any image can be converted into one of the “Convertible” image types.

AVDatabase application is specifically designed for creating audio/video databases by collecting real-time audio/video files via capturing from a peripheral video/audio device. An audio/video clip may include only video information, only audio information or both video and audio interlaced information. Several video and audio encoding techniques can be used with any encoding parameters.

Video can be captured from any peripheral video source (i.e. PC camera, TV card, etc.) in one of the following formats: YV12 (I420) à YUV 4:2:0, RGB24 (or RGB32) and YUY2 (YUYV) or UYVY [30]. If the capture format is other than YUV 4:2:0 then the frame is first converted to YUV 4:2:0 format for encoding. Capturing parameters such as video frame rate and frame size can be set during the recording phase by the user. The captured video is then encoded in real-time with the user-specified parameters, recorded into a supported file format and finally appended into the active MUVIS database. Video encoding parameters such as bit-rate, frame-rate, forced-intra rate (if enabled), etc. can be defined during the recording time. The supported file formats handle the synchronization of video with respect to the encoding time-stamp of each frame.

The framework MUVIS is developed for personal computers of work stations. However, it provides robust functionality of searching and retrieving the multimedia content. Furthermore, it implements the format conversion techniques for images which can be used to convert the image formats on the server side for mobile devices.

3.3. Mobile-MUVIS

Mobile-MUVIS (M-MUVIS) is a client-server framework where the client application is used to initiate the content-based query operation and send it to the server, which in turn performs the content-based query operation and sends the query results back to the client. Since Java is device agnostic, the client application developed in Java is therefore supported by a vast majority of mobile devices.

The Framework of M-MUVIS is shown in Figure 3, where the client application is used to initiate the content-based query operation such as (QBE) [18] and send the query request to the server, which performs the query operation and sends the query results back to the client. As shown in Figure 3, there are two servlets (web applications) on the M-MUVIS server side: the MUVIS Query Servlet MQS is used for performing content-based query operation, while the multimedia retrieval system (MMRS) is used for the media retrieval operation. As shown in Figure 3 query image feature can be extracted online (while performing the query operation) whereas audio and video clips features are extracted offline. As it may take a very long time to extract features from audio or video clips due to their unknown duration; it is not recommended to extract features while performing query operations.

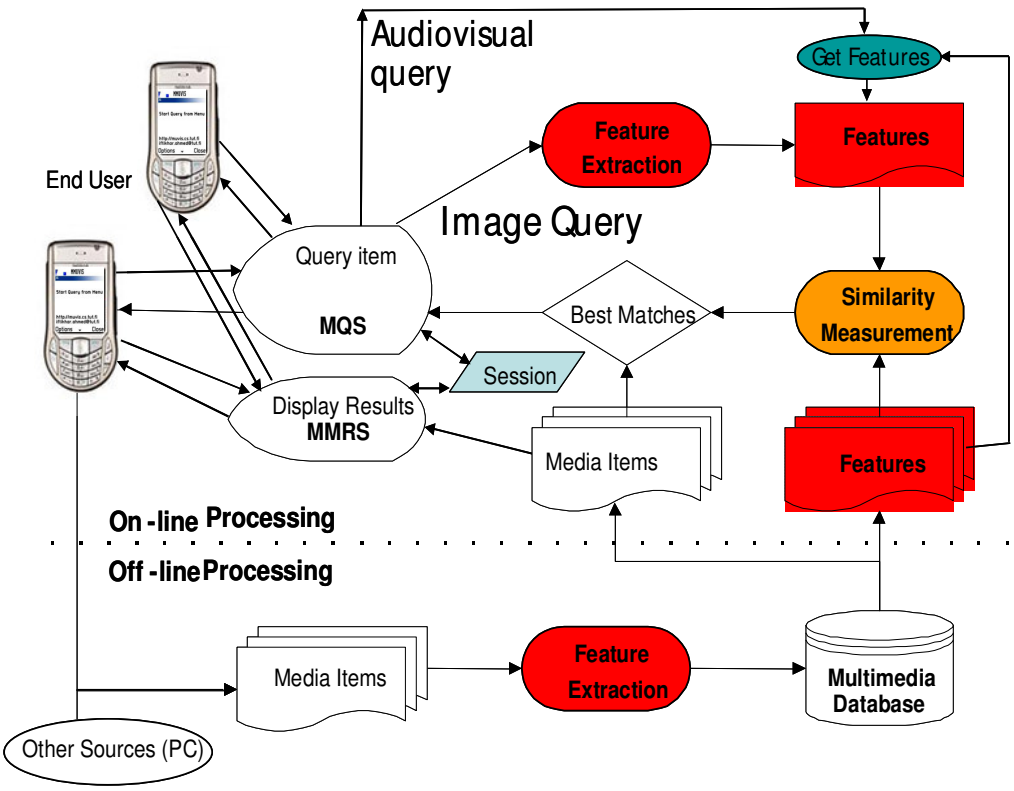

Figure 3: M-MUVIS Framework [24]

In order to perform a content-based query operation, the usual approach is to map the database items such as images, video and audio clips into some high dimensional vector space called feature domain. The feature domain may consist of several types of features extracted from the visual and audio content. Careful selection of the feature set to be used for a particular application is a key success factor in a Content-Based Media Retrieval (CBMR) system. Assuming that these features capture the semantic content of the media items; the perceived similarity between two items can then be estimated by the (dis-) similarity distance between their feature vectors. Therefore, the similarity-based retrieval problem with respect to a given query (item) can be transformed into the problem of finding database items whose feature vectors are close to the query feature vector. The exhaustive search based QBE operation is called Normal Query (NQ), and works as follows: using the available features of the query multimedia item and all the database items, similarity distances are calculated and then combined to obtain a unique similarity distance per database item. Ranking the items according to their similarity distances (to the query item) over the entire database yields the query results. This is done entirely on the server side.

M-MUVIS client application written in Java has a better control on the device resources such as camera, Bluetooth, and so on. Another advantage with Java is that the latest version of Java on mobile device supports hotspot, i.e. conversion of Java byte code to native code before execution. It takes the advantage of portability (machine independent) and at the same time uses the native code to deliver the best performance. To take the advantage of the flexibility and portability of Java on mobile devices, the M-MUVIS client has been developed using Mobile Information Device Profile (MIDP) [31]. MIDP is a Java profile that is supported by a wide range of mobile devices. It provides low and high level graphics widgets for UI. A Mobile device user installs the M-MUVIS client as a MIDP application on his mobile device.

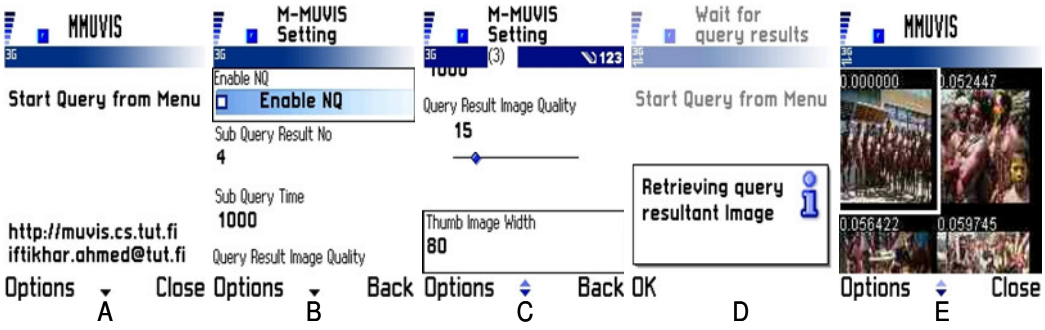

Figure 4: M-MUVIS client user interface on Nokia 6630. a) Main UI of M-MUVIS client b) and C) shows M-MUVIS setting view d) Query result retrieval wait dialog is shown e) Shows query resultant image. [24]

MIDP provide means to place commands [31] (for different actions, e.g. Ok, Back, Cancel, etc.) according to the device look and feel. M-MUVIS client is using a combination of high and low level graphics widgets for adapting the UI according to the device look and feel. Although it is also possible to create a unified UI by using low level graphics widgets that improve the user experience across a range of devices. Although it will not only consume extra memory (RAM) and processing power of the device but this unified UI might not be compatible with the look and feel of the device where application will run that can reduce the application usage by novice users. Users of an application should not spend more time in learning the UIs rather us it. Therefore, users of M-MUVIS client does not need to learn new UIs. M-MUVIS client uses the same look and feel as all the other applications on the device. Figure 4 shows the M-MUVIS client UI on Nokia 6630 [32].

M-MUVIS provides completely functional solution for the content based media retrieval to mobile devices. However, the Java based client application of M-MUVIS is needed to be installed on user's mobile device, which is not fairly easy for a normal mobile phone user. Another issue that is constantly raised is about the Java platform is performance. The portability of Java is also a major disadvantage, as byte code must always undergo some form of conversion so it can run on the native instruction set of the underlying architecture. The feature rich demands of next generation Java applications will quickly outstrip the capabilities of the current mass-market Java handsets. Hardware graphics accelerators, increasing processor clock speeds and fast data transfer rates are all changing the types of applications that can be run on mobile devices. If Java is to keep pace, then performance of the Java platform needs to improve and a powerful Java Virtual Machine (JVM) is required.

The solution provided by M-MUVIS is a robust solution to avoid the limitations of mobile devices. However, it still requires the phone to be Java enabled. Furthermore the client application is needed to be installed on the mobile device. And since common mobile device user is not accustomed to install software applications, M-MUVIS does not appear to be very user friendly.

4. Java Server Pages on MUVIS

In this section we propose a system that can process the multimedia content on powerful computer before it is delivered to a particular mobile device, and since in most of the cases the size of multimedia content will be reduced before it reaches mobile device, it minimizes the transfer of data over the network as well.

We present an efficient CBIVR framework, which uses combination of low-level features for image and video retrieval as compared to previous retrieval schemes, which only work over a single feature for content-based retrieval [33]. The proposed framework, Java Server Pages on MUVIS (JspMuvis) is a client-server architecture, where server is active on Personal Computer (PC) and an Internet browser application acts as client on the mobile devices.

JspMuvis is a web application that can be accessed using normal internet browser on any mobile device. It is based on contemporary Mobile-MUVIS (M-MUVIS) framework [24] but JspMuvis extends M-MUVIS in a way that users can now use the Internet browser on a mobile device for initiating a content-based query operation along with displaying the retrieval results, and Java is not required on the client side at all. Therefore, such a system provides a more generic solution, particularly for the mobile phones without Java being enabled. In order to retrieve the related device information (Internet browser name and version, screen resolution, screen size etc.) a standard approach is needed to present the device capabilities. Recently two new compatible standards have been created for describing delivery context based on the Resource Description Framework; CC/PP [14] created by the W3C and User Agent Profile (UAProf) created by the Open Mobile Alliance. JspMuvis uses DELI libraries to extract the information (User Agent Profile) about the connected mobile devices and generates the device specific content.

The solution provided by JspMuvis resolves the java dependency as well. Since all the processing for CBIVR is being done on server side, which is a powerful computer, the only functionality to be performed on mobile devices should be display of the content. As mentioned earlier, display of content on the small screens and with limited capability browsers is still not mature enough. Large images are not useful on mobile devices, and those larger than screen will make for a worse user experience. The images shouldn't be larger than about half screen size (about 150-100 pixels). Opera in Small-Screen mode will not display many images and resize others to fit the screen. However, with the help of Java Server Pages (JSP) [34] we can generate Extensible Hyper Text Markup Language (XHTML) pages, which are already tailored for different user interfaces for different devices. Dynamic selection of Images and Videos is generated

according to the size and capabilities of the device. Also the content provider must have one interface which is capable to adopt according to the capabilities of the device.

Different technologies and standards are available to define the capabilities of any mobile device. Also libraries have been developed to receive these capabilities and feed them to any specific application. JspMuvis uses User Agent Profile [12] as a source of the mobile device capabilities and Deli libraries to resolve the HTTP requests and gather the UAProf information. In this chapter we explain the functionalities and technical details of JspMuvis and chapter 5 to chapter 7 explain the technologies that have been used by JspMuvis.

4.1. JspMuvis Functionalities

As mentioned earlier JspMuvis is extension of the research, in the field of content based multimedia search and retrieval, going on at Tampere University of Technology (TUT) [19]. The directed objective of JspMuvis is to search, retrieve and display images and videos on mobile devices, and display images and videos on any mobile device's screen in an optimal way. JspMuvis extends MUVIS project in such a way that all of the MUVIS end user features are available on mobile devices.

Since most of the mobile devices are equipped with internet browser, the main requirement for JspMuvis is that user is able to use internet browser to search and retrieve images and videos. User is able to provide the image, which exists on mobile device, and send a search request to JspMuvis running on a web server, and the search can be performed on image and video databases. However, it is not necessary for user to provide the query image to see the content based search and retrieval in operation. JspMuvis is capable of selecting a random image from the image database and sending the results to mobile device in a customized format. The query results clearly show the query image and its' matching results. It also shows the match percentage.

After the results are displayed on mobile device's screen, user can select any image and retrieve the image in its' original form.

The query image can be searched in video database as well. In this search image is matched against each frame of every video present in the video database and the results are given to the mobile device in a customized fashion. JspMuvis can operate on two kinds of databases; Image Database (contains only images) and Hybrid Database (contains images and videos). An image query in image database retrieves only images and an image query in hybrid database can retrieve images as well as videos.

For every single query the resultant images are formatted and resized to fit the specific mobile device screen and as a result they appear as thumbnails to the end user. Each media is retrieved over Hypertext Transfer Protocol (HTTP) link and multiple copies of media are not made.

For every query, the resultant thumbnail images are kept on file system on server computer and their location information is stored in an XML file. This makes it easier to retrieve these images at any time.

Many of the mobile devices do not support faster networks (3G, WLAN etc.) and retrieval of images to those devices can be very slow, especially if there are a number of resultant images. To resolve this problem Progressive Query (PQ) is implemented. PQ sends the resultant image to the mobile device as soon as it is processed and doesn't wait for remaining images to be processed. Hence the resultant images are continuously sent to the mobile device and they are displayed in sequential order. This approach reduces the waiting time for the end user.

Once the resultant images are displayed on mobile device, it is also possible to chose any of the resultant images and perform a content based search, in this way the resultant image becomes the query image for new search query. This operation is called 'Find Similar'.

4.2. Proposed System

This section contains the overview of the JspMuvis along with its evident features explaining how it fulfils the needs of displaying web based images and videos on any connected mobile device's screen in an optimal way. As discussed earlier the best approach to display customized web images and videos would be to do the customization on powerful computers before sending the content to mobile device. Since most of the mobile users use internet browsers to surf the internet, JspMuvis proposes the implementation that customizes the images and videos on internet server which is running on powerful computer.

This project is an extension of the framework, MUVIS and M-MUVIS. In JspMuvis the media (images and Videos) can be searched and retrieved on the bases of their contents. JspMuvis application is running on TomCat [26] web server which resides on a computer connected to internet. The search, retrieval and customization of images and videos is done on server side. The internet browser on mobile device acts as client to JspMuvis server. Figure 5 shows the client server architecture of JspMuvis.

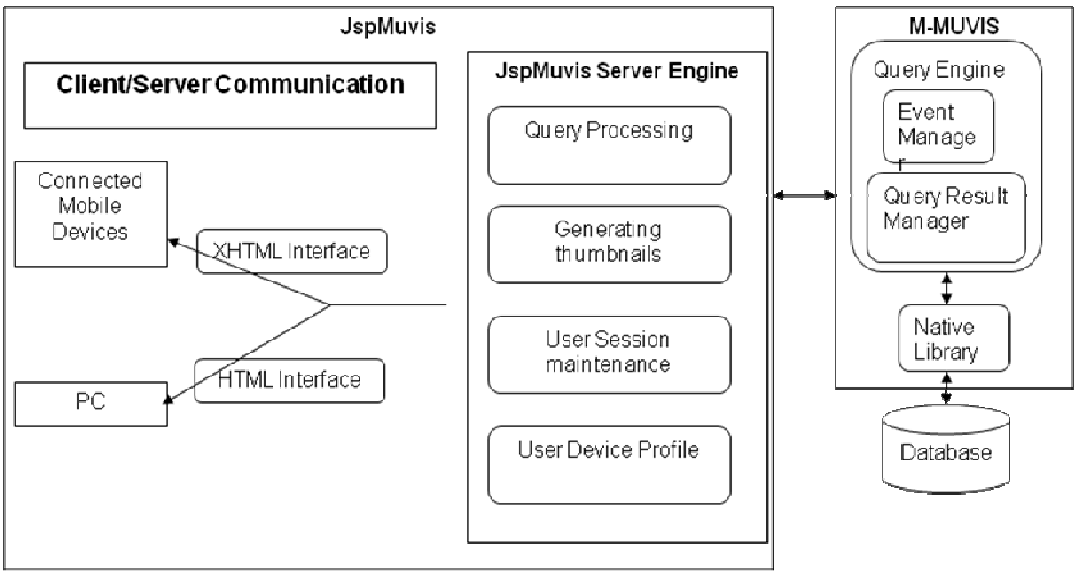

Figure 5. Proposed System

Java Server Pages (JSP) architecture has fulfilled the overwhelming need to simplify the web application's design by separating the dynamic content from static template display data. This architecture is a layered approach and we place each logical part of the application on right layer.

The access and use of a web browser is quite habitual and periodic these days and secondly many mobile devices might not support interpreted languages such as Java. Therefore JspMuvis uses markup language, e.g. XHTML with device browser for CBMR. In JspMuvis, the server generates XHTML pages dynamically for the mobile devices. Wireless Application protocol (WAP) is a standard for applications (internet browsers etc.) that use wireless communication. The main purpose of WAP is to enable communication to the Internet from a mobile device. Mobile devices can use a Web or WAP browser to display those XHTML pages. JspMuvis server (TomCat [26] web server) runs on a computer. As most of mobile device users are novice users and not computer programmers. Furthermore, their mobile devices are for their personal usage. Therefore they are reluctant to install new applications. In M-MUVIS a user has to install a Java application on the device to perform the CBMR; whereas, in JspMuvis they do not need to install any application on their devices, XHTML pages are downloaded as in normal browsing and used for CBMR.

JspMuvis running on Nokia N90 is shown in Figure 6. Query Image is shown in Figure 6 (A), this image is uploaded via web browser to JspMuvis server for CBMR. JspMuvis query form is shown in Figure 6 (B). After content based search and retrieval results are formatted according to the mobile device capabilities. Figure 6 (C) to (E) shows the search results which are matching images in customized. JspMuvis also provides the functionality to retrieve the image in its' original form. Figure 6 (F) shows one of the resultant image in its' original form. For every query resultant image (QRI)

there exists an option to take the QRI as query image (QI) and perform content based search. Figure 6 (G) shows the 'Find Similar' option highlighted and Figure 6 (H) shows the query results of find similar operation

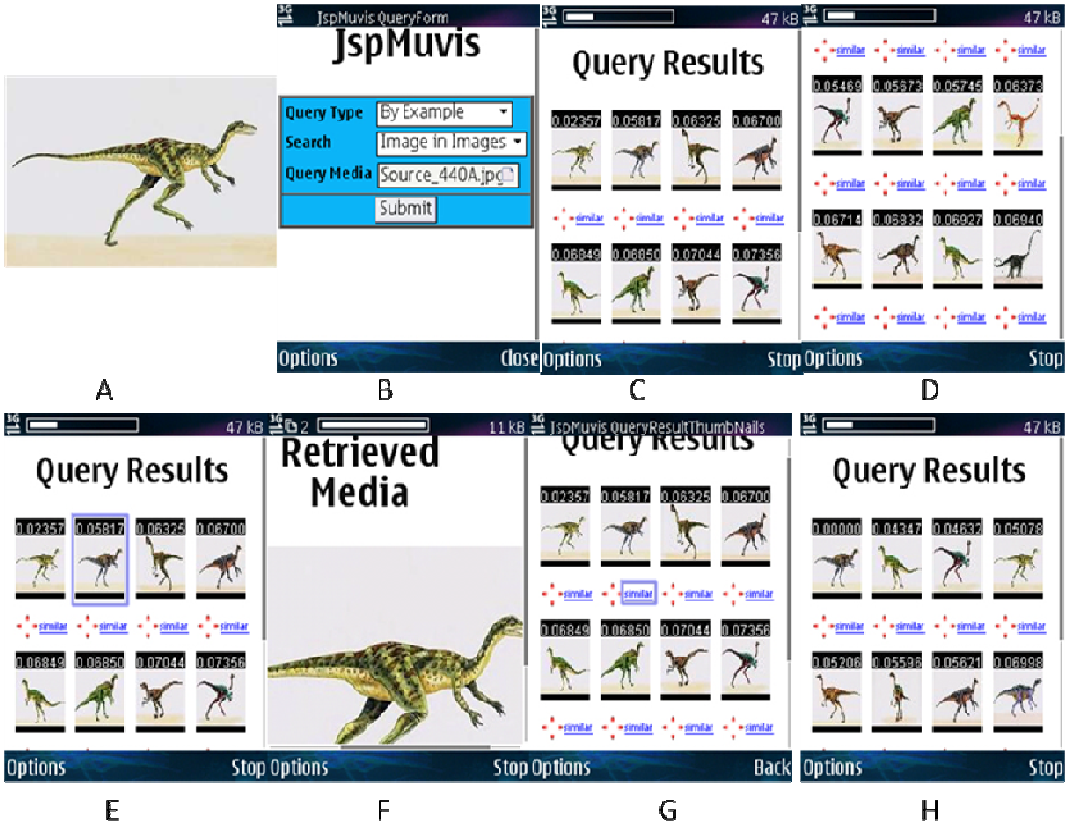

Figure 6. JspMuvis user interface on Nokia N90

Query operation and media retrieval (optionally conversion in different format) should be independent from each other such that one operation should not block the other operation. JspMuvis server comprises java beans [25] running inside a Tomcat web server which in effect performs the content-based query operation and media retrieval on the server side. The Java beans in JspMuvis make the query and media retrieval two independent operations.

JspMuvis server side application performs content based search of the selected image. As soon as the user submits a request, it is sent to JspMuvis server. JspMuvis server receives and identifies request. It detects the 'Query Type', 'Search Type' and receives image given in the field of 'Query Media'. The selected image is uploaded to the server of JspMuvis for content based search. It is done by the automatic communication between JspMuvis form, which user sees, and the server side application. The request sent by mobile device to the JspMuvis is an HTTP request [10] which also contains the user agent [11] information of the device. In order to receive the mobile device capabilities information JspMuvis uses User Agent Profile [12] that enables mobile

devices to declare their capabilities to web servers and web services. As soon as JspMuvis server application is done with the analysis of request from the mobile device it starts the content based search in the database. After the search completes JspMuvis formats the resulted images according to the device information. The formatting of the results is a critical part of JspMuvis where every resulted images resized in a way that it would fit to the screen of mobile device along with the other images. JspMuvis sends response to the request initially received from mobile device. This response contains the results of the content based image search which are then displayed to the screen of mobile device as shown in Figure 7.

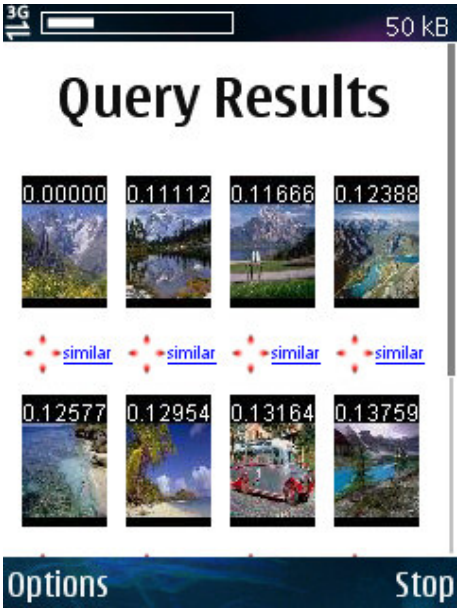

Figure 7. Query Results

4.3. JspMuvis Architecture

In the proposed framework JspMuvis server generates XHTML pages dynamically for the mobile devices. Mobile devices can use a Web or WAP browser to display those XHTML pages.

The communication between different components of JspMuvis is shown in Figure 8. QueryForm.jsp receives the first request from the user. It uses Deli Library via DisplayProfile.java to retrieve the device capability information which is then kept stored in the instance of DisplayProfile.java for the whole user session [25]. QueryForm.jsp displays the first screen containing the selection form. The output of QueryForm.jsp is the form to specify query criteria, as the example in Figure 6 (B). QueryProcessor.jsp is the main controller to serve user requests. It uses separately implemented functionalities of JspMuvis to serve the user request. The functionalities used by QueryProcessor.jsp are implemented Java Beans [29] shown in violet color boxes.

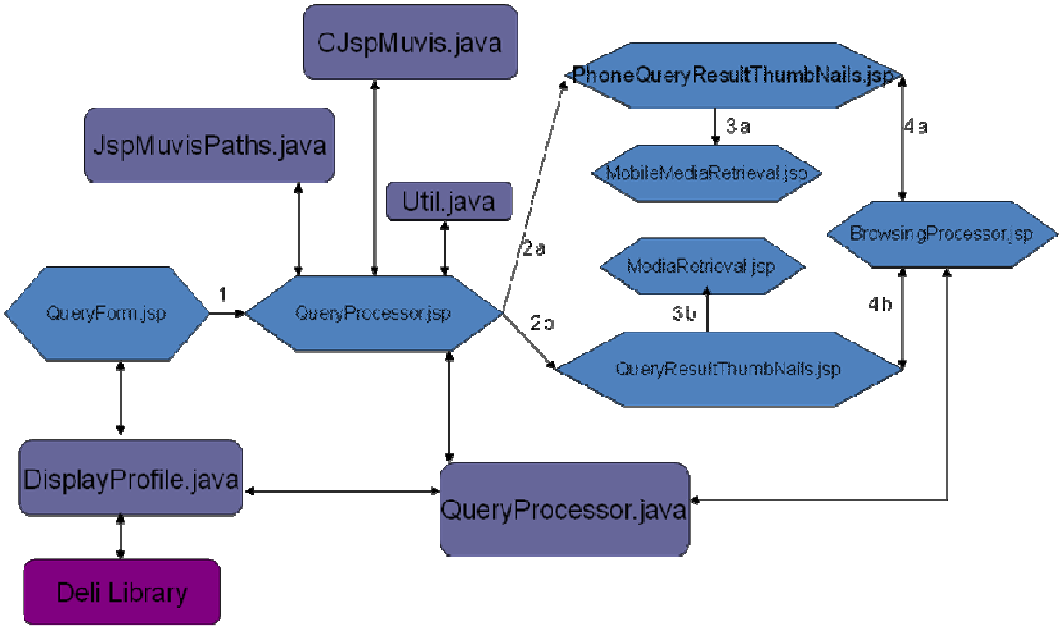

Figure 8. JspMuvis Flow

QueryProcessor.java contains the main functionality of formatting the retrieved images and videos according to the mobile device capabilities stored in the instance of DisplayProfile.java. It is responsible for communication with MUVIS framework to search and retrieve the multimedia content.

JspMuvisPaths.java is a Java Bean that is responsible for the system configurations. System configurations make JspMuvis portable and it can easily be deployed on any web server. QueryProcessor.jsp uses the path 2a for all the phone devices, but if the user is using personal computer it chooses path 2b since the customization of the retrieved multimedia content is not needed.

The customized and formatted results of content based image or video query are displayed using XHTML on BrowsingProcessor.jsp.

As stated earlier JspMuvis has client-server architecture. JspMuvis server (Tom Cat web server [26]) is kept active on a powerful computer. The client and the server communicate over Hypertext Transfer Protocol HTTP [10]. HTTP is a stateless protocol so a session [25] is created on the server side to keep track the client information (query type, mobile device screen size, etc.). The overview of JspMuvis framework is shown in Figure 9. The first operation done by JspMuvis is to parse the user request and extract the device capability information from User Agent Profile using Deli Libraries. The second step is to perform the content based query. In the third step the results are formatted according to mobile device's capabilities and are displayed on mobile device.

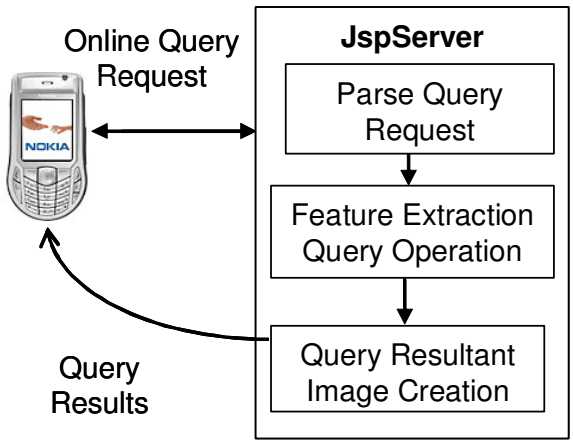

Figure 9. The Overview of JspMuvis Framework

In JspMuvis, session is an object created and maintained by Tom-Cat web server on the server side. Session for each user is created on first receiving the request as shown in Figure 10. Session persists across all connections from client to server till the user terminates the Internet browser (i.e. the client application) on the mobile device. Server assigns a unique identification number to each session and passes that number to the client. The client uses that number for each request of established connection with the server. The server can therefore identify a particular client and uses its session information for any requested operation. In JspMuvis, one client can have only one session, after the completion of a content-based query operation, the retrieval results are also integrated into the session stream in case any client can request these results later.

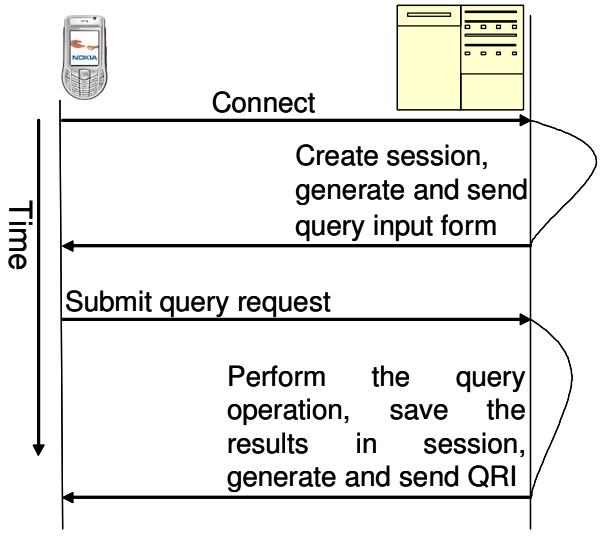

Figure 10. JspMuvis client-server communication

There are mainly three layers in JspMuvis.

1. Presentation Layer

This is the user interface layer of *JspMuvis*. XHTML pages that are generated by the JspMuvis server contain the user input form and the option settings for a content-based query operation described below. Presentation layer is responsible for interacting with the user. The user has the following options in this layer as one particular example displayed in Figure 11.

- **Query Type**: Selection for either "Random Query", or "Query By Image".
- **Search**: Selection of the query type. Image in Images, Image in Videos, or Video in Videos
- **Query Media**: User can provide a query image in case "Query by Image" option is chosen.

In a "Random Query" operation, query is performed after picking an image randomly among the images in the active database on the server side. "Query by Image" is a typical QBE process where the user selects a particular image (via "Query Media" edit box in Figure 11) and initiate a query operation. The query image is then uploaded to the server, the query process is performed and the retrieval results are streamed to client and shown to the user in an Internet browser as one particular example is shown in Figure 12.

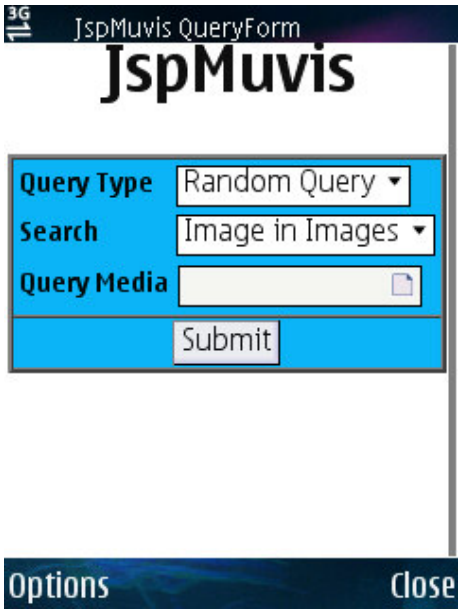

Figure 11. User settings for content-based query on Nokia N95

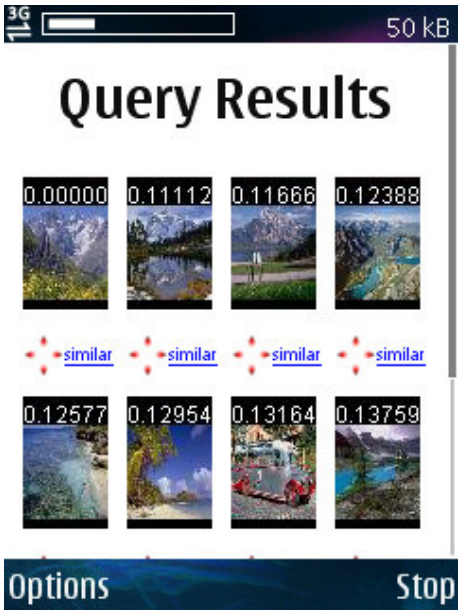

Figure 12. A sample QRI in an Internet browser on Nokia N95

The presentation layer is also responsible for displaying the query results. Considering the small screen size of mobile devices query results are displayed as thumbnails. A "Query Resultant Image" (*QRI)* is created on the server side where image thumbnails of the retrieval results are drawn. An example *QRI* is shown in Figure 12.

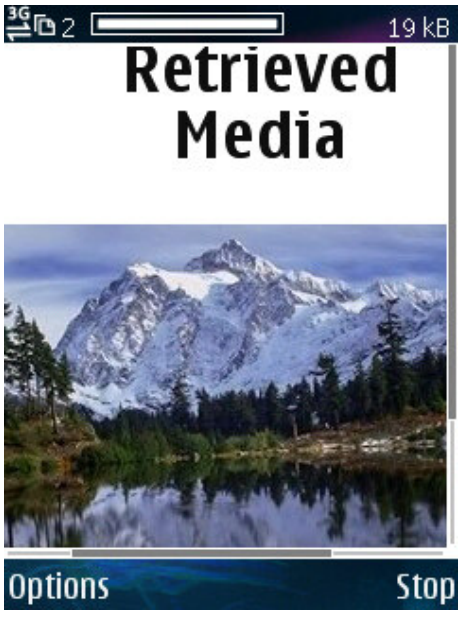

Figure 13. Retrieved image shown in an internet browser on a mobile device

The user can select any thumbnail among the images in a *QRI* and the client can fetch the original image (with real dimensions) from the server. Figure 13 shows the retrieved original image that was the second thumbnail in the first row of results in figure 13.

2. Application Layer

In this layer the necessary coordination between presentation layer and data layer is organized. Application layer is responsible for using the data layer according to the user selections in presentation layer. It captures the query information from user input form via presentation layer, uploads it with the selected image to the *JspMuvis* server and uses the data layer to perform the query operation. The critical part of customizing the query results according to client mobile device is done in this layer.

As mentioned earlier, a session is created between client and server. Application layer is also responsible for session tracking of a particular *JspMuvis* client. It uses cookies [25] to store the session information on the mobile device, and Internet browser picks the session information during the beginning of a client session.

3. Data Layer

Data layer is responsible mainly for database operations. *JspMuvis* uses the APIs developed in *M-MUVIS* project to access the database through native (C/C++) code. Native libraries are used for efficient content-based image query related operations. *JspMuvis* contains Java beans [26], which handle the operations such as activating the selected database within application, performing the content-based query operation, retrieving the 12 best results embedded into QRI.

5. User Agent Profile and Composite Capabilities/Preference Profiles

Web applications accessed from traditional HTML browsers running on desktops make assumptions about such client capabilities as screen size, bandwidth, support for color images, and so on. These assumptions break down when the same content is accessed from mobile devices, whose capabilities are more limited and varied. The challenge for application developers is to support thousands of mobile devices with widely varying capabilities. Customizing content for different devices, and for different users, requires significant investments of time and effort.

One way to ensure compatibility among the largest set of devices is to settle for the least common denominator, but then users of high-end devices are limited to the capabilities of lower-end devices, and there is little scope for user preferences. The challenge is then: How do you deliver content that reflects users' preferences and the capabilities of their devices without the time and effort of tailoring the code to each platform?

Composite Capabilities / Preference Profiles (CC/PP) [14] and User Agent Profile (UAProf) [12] are the standards developed by the World Wide Web Consortium (W3C) and Open Mobile Alliance (OMA) respectively, aiming at providing a structured format to describe device capabilities and user preferences for the purpose of adapting contents to a device. CC/PP is a general framework that defines the structure of a vocabulary (that describes device capabilities and user preferences). UAProf is a specific vocabulary based on CC / PP. Valid UAProf 2 [12] profiles are also valid CC/PP [14] profiles as the CC/PP model follows UAProf.

5.1. Resource Description Framework

The Resource Description Framework (RDF) is the W3C foundation for processing metadata i.e. information about information. RDF is a W3C recommendation since February 2004 [35]. It aims to provide interoperability between applications that exchange machine-understandable information on the Web. RDF provides a model for data, and syntax so that independent parties can exchange and use it. This framework is designed only to be read and understood in between computers and not for being displayed to the people. The documents of Resource Description Framework are written in XML format and the language used by RDF is called RDF/XML.

RDF provides the framework with the basic tools for both vocabulary extensibility, via XML namespaces [36], and interoperability. There is a specification that describes how to encode RDF using XML, and another that defines an RDF schema description language using RDF [37].

Essentially RDF models consist of a collection of statements about resources. A resource is anything named by a URI plus an optional anchor ID e.g. in "http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppschema-20021212#HardwarePlatform" The URI is everything before the hash and the anchor ID is everything after the hash. An RDF statement comprises of a specific resource together with a named property plus the value of that property for that resource. These three individual parts of a statement are called, respectively, the subject, the predicate, and the object. The object of a statement can be another resource or it can be a literal i.e. a simple string or other primitive data type defined by XML.

5.2. Composite Capabilities / Preference Profiles

CC/PP define a general purpose structure for a profile, which contains a set of attributes about the device capabilities and user preferences of a device, but it does not define any specific attributes. In other words, CC/PP provide the rules of how to construct a vocabulary that describes capabilities and preferences, but does not specify the actual attribute names and values [14].

CC/PP is based on RDF [35], which is a language used to represent metadata about web resources in a machine understandable format. A CC / PP profile contains one or more components, which are categories of features. Each component contains one or more attributes. Each component may optionally specify a set of default values for the attributes it contains. CC/PP allow a client (a device) to describe its capabilities by referencing a standard profile and to make additions or modifications to the standard profile.

CC / PP is intended to be extensible. As CC/PP define the structure of vocabularies, different vocabularies based on the same structure can be created for different purposes or applications. A profile can also be assembled from parts of several vocabularies. Therefore new components and attributes can be created and existing components and attributes can be re-used to create a new CC/PP expression.

A CC/PP profile is a description of device capabilities and user preferences that can be used to guide the adaptation of content presented to that device. Also the World Wide Web Consortium (W3C) has finalized the CC/PP [14] standard for representing device capabilities and user preferences. Java specification request (JSR) for CC/PP can be found from appendix E.

A CC/PP profile is broadly constructed as a two-level hierarchy: a profile has a number of components and each component has a number of attributes. The attributes of a component may be included directly in a profile document, or may be specified by reference to a default profile that may be stored separately and accessed via a URL. CC/PP distinguishes between default and non-default values attributes such that non-default values always take precedence.

Although a CC/PP profile is a two level hierarchy, it is commonly represented using an XML serialization of an RDF model. Crucially the underlying RDF model describing a profile is more complicated than a two level hierarchy. This can be demonstrated by processing a profile using the W3C RDF validation service referenced in the previous section. Some examples of these complexities are as follows: Firstly simply giving a component a standard name (e.g. Hardware Platform) is not sufficient to distinguish it as a particular component. In addition it must have an rdf:typeproperty that indicates it is an instance of a particular component type in a particular namespace.

A CC/PP profile is a description of device capabilities and user preferences that can be used to guide the adaptation of content presented to that device. Here profile does not refer to a subset of a particular specification, for example the CSS Mobile profile, but refers to the document(s) exchanged between devices that describe the capabilities of a device.

As the number and variety of devices connected to the Internet grows, there is a corresponding increase in the need to deliver content that is tailored to the capabilities of different devices. Some limited techniques, such as HTTP 'accept' headers and HTML 'alt=' attributes, already exist. As part of a framework for content adaptation and contextualization, a general purpose profile format is required that can describe the capabilities of a user agent and preferences of its user. CC/PP is designed to be such a format.

CC/PP is based on RDF, the Resource Description Framework, which was designed by the W3C as a general purpose metadata description language. RDF was designed to describe the metadata or machine understandable properties of the Web. RDF is a natural choice for the CC/PP framework since user agent profiles are metadata intended primarily for communication between user agents and resource data providers.

A CC/PP profile contains a number of CC/PP attribute names and associated values that are used by a server to determine the most appropriate form of a resource to deliver to a client. It is structured to allow a client and/or optionally a proxy to describe their capabilities by reference to a standard profile, accessible to an origin server or other sender of resource data, and a smaller set of features that are in addition to or different than the standard profile. A set of CC/PP attribute names, permissible values and associated meanings constitute a CC/PP vocabulary.

It is anticipated that different applications will use different vocabularies; indeed this is needed if application-specific properties are to be represented within the CC/PP framework. But for different applications to work together, some common vocabulary, or a method to convert between different vocabularies, is needed. (XML namespaces can ensure that different applications' names do not clash, but does not provide a common basis for exchanging information between different applications.) Any vocabulary that relates to the structure of a CC/PP profile must follow this specification.

5.3. User Agent Profile

A user agent is a program, such as a micro-browser on a mobile phone that acts on a behalf of a user to interact with some information server. A user agent profile contains a set of characteristics related to the device capabilities and network configurations, and is to be transported along with a request over the mobile network and the internet to an information server for content formatting purposes [38].

User agent profile (UAProf) is a structure that contains the mobile device information. UAProf 2.0 [12] is an OMA specification that is designed to allow wireless mobile devices to declare their capabilities to data servers and other network components. The design of UAProf is based on Resource Description Framework (RDF) [37].

The User Agent Profile (UAProf) specifications define a specific vocabulary (components and attributes) to describe the capability and preference information. UAProf also describes the flow of a user agent profile from a client (a device) to a server based on the WAP model.

Each user agent property is defined as belonging to one of a small number of components, each of which corresponds to an aspect of a user agent device. UAProf contains six components to describe capability and preference information. These include:

- Physical components of a device, Hardware Platform, e.g. vendor, model, screen size, input and output methods.
- Applications and programs available on device, Software platform, e.g. operating system, java support, audio and video encoder.
- Communication capabilities of the device, Network characteristics, e.g. security support, bearer support (bearer is the lowest level protocol in the WAP architecture).
- Inter Browser user agent, e.g. browser name, supported HTML and XML versions.
- WAP characteristics, e.g. supported WAP and WML versions.
- Push characteristics, e.g. content types of a push message, maximum size of a push message.

When a new phone is about to release you will find it on the list of UAProfile. An example of the Nokia E65 UAProfile is available in appendix C. Notice that there are several profiles for the E65. From this XML file you get significant information about the mobile phone. Location of list of available profiles for different vendors can be found in appendix D.

When a request is sent from a client to a server, the user agent profile is attached to the request. The user agent profile (or the request) originates from a client and goes

through the mobile network and some proxy (or proxies) before it reaches the server. Since UAProf is based on WAP 2.0, the path along which the user agent profile travels is much the same as Figure 14.

At the origin, the message makes reference to a default user agent profile that contains the default values for the attributes of the particular mobile device and user agent. The default user agent profile is expected to reside in some repository to which the server has access. At each of the intermediate stages, the node may add or override the attributes in the user agent profile to indicate the characteristics of that particular node.

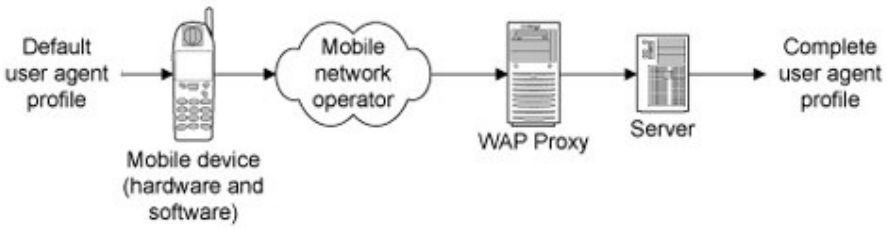

Figure 14. The Flow of a User Agent Profile

Specifically, the default user agent profile, to which the client initially refers, may contain information about the hardware platform, software platform and the browser on the device. When the profile goes through the mobile network, the mobile network operator may add or override the network characteristics in the profile. When the profile reaches a WAP proxy, the profile may be updated with WAP characteristics. The profile, along with the request, would finally reach the server, and would have been modified with characteristics of all the nodes it has passed through.

The server needs to parse the user agent profile and extract capability and preference information in order to return a response (some content) in a format that can be interpreted by the client. The server first obtains the default profile from a profile repository. It then applies all the added or overriding attributes specified by the intermediate nodes to the profile, thus generating a complete user agent profile. This process is known as profile resolution.

It is anticipated that there may be new hardware, software or other characteristics, or new applications that make use of these new characteristics in the future. Therefore there may be a need for new vocabularies to represent the new attributes. As UAProf is based on CC/PP, which follows the RDF model, it permits extensions to the base vocabulary, i.e., the UAProf vocabulary. The new vocabulary would follow the structure defined by CC/PP (note that CC/PP defines the rules to construct vocabularies) and would be represented by a new RDF schema that specifies the new resources and the associated properties. A profile can be constructed from multiple vocabularies (multiple RDF schemas) and therefore is able to include the attributes from the base vocabulary and the new attributes from the extension.

6. XHTML

The Extensible Hyper Text Markup Language [44] is an extension of HTML 4 [6] and is ultimately designed to work in conjunction with XML-based user agents.

XHTML 1.0 is the first document type in the XHTML family. It is a reformulation of the three HTML 4 document types as applications of XML 1.0 [7]. It is intended to be used as a language for content that is both XML-conforming and, if some simple guidelines are followed, operates in HTML 4 conforming user agents. Developers who migrate their content to XHTML 1.0 will realize the following benefits:

- XHTML documents are XML conforming. As such, they are readily viewed, edited, and validated with standard XML tools.
- XHTML documents can be written to operate as well or better than they did before in existing HTML 4-conforming user agents as well as in new, XHTML 1.0 conforming user agents.
- XHTML documents can utilize applications (e.g. scripts and applets) that rely upon either the HTML Document Object Model or the XML Document Object Model [45].
- As the XHTML family evolves, documents conforming to XHTML 1.0 will be more likely to interoperate within and among various XHTML environments.

The XHTML family is the next step in the evolution of the Internet. By migrating to XHTML today, content developers can enter the XML world with all of its attendant benefits, while still remaining confident in their content's backward and future compatibility.

7. Deli Libraries

HP labs have developed a set of libraries to provide functionality of resolving HTTP requests to get the delivery contexts of the client devices [39]. DELI is an open sourced java based library which gathers CC/PP or UAProf. JspMuvis uses DELI libraries to get the information on client mobile device capabilities and provides optimized content to different clients based on their capabilities.

DELI starts with the Http request from the client extracting the user-agent information. First DELI loads a namespace definition file (namespaceConfig.xml) that points to several local UAProf schemas. These schemas are corrected versions of the schemas published by the OMA [13].

In some cases the namespace definition file contains namespace alias information, because when some of the schemas were first published they were published from different URIs from those specified in the UAProf specifications. This caused confusion amongst UAProf profile authors about which URI was correct. Therefore if a profile uses a namespace specified in a specification (a known namespace), it is regarded as correct. If it uses a namespace that a schema was published from, but was not used in a specification (known as an aliased namespace) then DELI prints a warning about this but accepts the profile. If the profile uses properties that are not from either a known or aliased namespace, then errors will be printed when those properties are encountered.

A typical entry in the namespaceConfig.xml file is indicating to load a schema file, from the location indicated by <schemaVocabularyFile>, and associate it with the known namespace indicated by <uri>, and alias namespaces indicated by <aliasUri>.

```
<namespace>
<uri>http://www.wapforum.org/UAPROF/ccppschema-20000405#</uri>
<aliasUri>http://www.wapforum.org/profiles/ccppschema-20000405#</aliasUri>
<aliasUri>http://www.wapforum.org/profiles/UAPROF/ccppschema-
20000405#</aliasUri>
<schemaVocabularyFile>config/vocab/ccppschema-
20000405.rdfs</schemaVocabularyFile>
</namespace>
```

7.1. Load schemas

Then DELI loads each schema file. It extracts information about

- Namespace associated with a particular vocabulary
- Component types specified in the vocabulary
- Properties available in each component type

- Data types associated with each property
- Resolution rule associated with each property
- Whether each property is single valued, or can contain multiple values which are either unordered (a Bag) or ordered (a Sequence)

Prior to UAProf 2 [12], not all of this information is not contained in the RDF Schema, so it is necessary for DELI to process the comments field in the schema. There have been also been instances of inconsistencies between the information in the comments and information in RDF Schema, so even if the information is available in the RDF Schema, DELI still parses the comments and prints a warning when there is an inconsistency.

7.2. Parsing RDF / XML

Then DELI starts to check the profile. It first uses ARP [40], the parser used in Jena [41], also used in the W3C RDF validation service [42], to convert the UAProf document to an RDF model. This checks the document is conformant with RDF/XML [43], just as the W3C validator does. If the document does not conform to UAProf specifications then errors are printed. There are some common errors because some (now obsolete) UAProf specifications were written before RDF/XML Syntax Specification were written, so they feature an older, obsolete version of RDF. Typical examples here include the use of the id attribute instead of the rdf:id attribute or about rather than rdf:about.

7.3. Processing RDF model

Then DELI checks the RDF model derived from the profile. It does this as follows: First it locates the root of the profile. It expects the root node will have either uaprof:component properties or uaprof:default properties that have uaprof:component properties. In either case, the uaprof:component properties will point to an anonymous node that has an rdf:type property and a number of UAProf properties. DELI checks that the type property corresponds to a component type defined in one of the schemas loaded in stage 1.2 (i.e. it uses the same local name and namespace). It then proceeds to check each UAProf property attached to the node.

In the case of all profiles, it checks that the property corresponds to a property defined by a schema in stage 1.2. If not, it prints an error at this point. It checks that the property is attached to the correct component. If not, it prints an error at this point. The schema also defines whether the property should be a single value or a Bag or Sequence so DELI checks this is true in the profile. If not, it prints an error at this point.

In the case of non UAProf 2 [12] profiles, if data type validation is turned on (it is by default) then it also checks that the property value conforms to the regular expression defined in the uaprofValidatorConfig.xml file.

In the case of UAProf 2 [12] profiles, it checks that the property has an rdf:datatype attribute and that it corresponds to the data type defined in the schema. It also checks the property value conforms to the regular expression defined in the XML Schema file associated with the vocabulary, which is currently the xmlschema-20030226.xsd.

Because DELI checks the RDF model derived from the profile, the ordering of components and properties do not correspond to the ordering used in the original UAProf document. This is an unfortunate side-effect of using RDF. However DELI users would like line numbers to be printed with errors, so my intention is to investigate whether DELI can re-analyze the UAProf document once an error has been determined in order to determine a line number.

7.4. Summary

In summary, DELI checks that:

i) A profile is well formed RDF/XML.

ii) Correct RDF namespace is used

iii) Profile only uses known components. This is based on the assumption that unknown components are mis-spelt components.

iv) Profile only uses known properties. This is based on the assumption that unknown properties are mis-spelt properties

v) Each property is associated with the correct component

vi) A property is single valued, sequence valued or bag valued as indicated by the schema

vii) The property value conforms to a regular expression for the data type associated with the property in the schema in the case of UAProf 2 [12] profiles,

viii) The property has an rdf:datatype attribute containing the correct data type for the property.

8. The experiments

This section describes the test cases and experiments done to check the capabilities of the system. The system is tested on different mobile devices and different networks. We shall test if the user can make a content based search for an image from his/her personal photo selection on the mobile device. And confirm that the results of this content based search are retrieved and displayed to the mobile device in an optimal way. The experiments should reveal that the images in search query results are formatted according to specific mobile device capabilities and are structured to fit into the mobile device's screen in a harmonized way.

Since one part of this research was to display web images in an optimal way, the response time of each operation should be calculated. we shall also calculate the amount of time taken by a search query on different networks.

8.1. Display of Query Results to Mobile Devices

This experiment evaluates how the results of content based search of an image are displayed on two different mobile devices, hence expecting the outcome of the practical work of this thesis. Content based search for an image can produce number of resultant images. And since, most of the captured images are larger than an average mobile device's screen, JspMuvis should be able to fit all the results in a well formatted structure to the mobile device' screen. This experiment evaluates the display of a large image after customization to a mobile device. A mobile device is needed which is connected to internet and is running in internet browser. We chose two mobile devices, Nokia N95 and Nokia N81, for this experiment which fulfill the basic needs of a normal internet user.

The display of Nokia N95 is 240x320 pixels, we chose query image of the dimensions 384 * 256 Pixels shown in Figure 15. The JspMuvis server runs on a PC equipped with Pentium 4, 2.99GHz and 1.9GB of RAM and the database contains 1000 images of different types and sizes. Nokia N95 uses 3G network to access internet.

Figure 15. Query image for experiment: Display Query results to Mobile Device

The options selected in the JspMuvis form are, Query by Example, search image in images, and the path to query item is provided as in Figure 16.

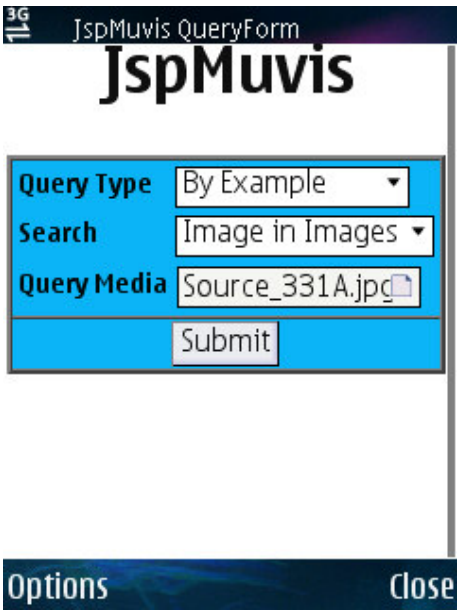

Figure 16. Search Options set to JspMuvis query form

After pressing the submit button, HTTP request is sent to JspMuvis server along with the device information and the query image is uploaded to the JspMuvis server. Content based search is performed to the image database and JspMuvis customizes the query results according the device information of Nokia N95. The HTTP response received at the client side (Nokia N95) contains the XHTML page which is capable of

displaying the query results according to the device capabilities. Figure 17 shows the query results displayed on Nokia N95 screen.

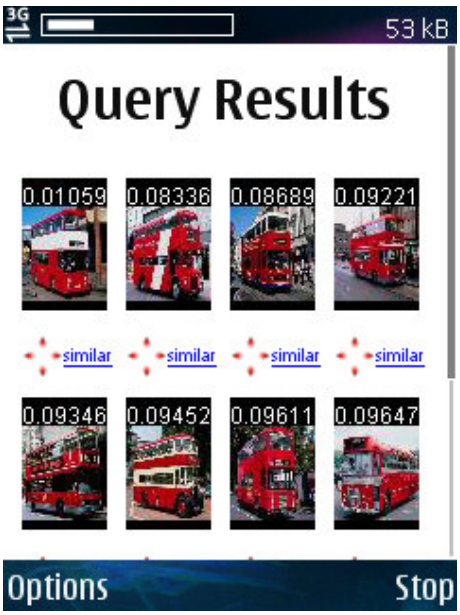

Figure 17. Results of content based query on Nokia N95

The images found as the result of query are of different sizes, quality and formats. JspMuvis formats and resizes first 12 resultant images in a way that they are easily displayable on the 240 x 320 Pixels sized screen. As you can see from the Figure 17 that the resultant images are displayed properly on the small screen and the user does not have to scroll horizontally. Vertical scrolling however exists.

We extend the experiment to evaluate the extended feature of JspMuvis, this feature is to make content based search from within the query results. Figure 17 also shows that each resultant image has an option ‘similar’ below it; this option provides the functionality to perform QBE using the particular QRI as QI. We select second QRI to see the ‘Find Similar’ functionality of JspMuvis as shown in figure 18.

Figure 18. Find Similar feature of JspMuvis

JspMuvis performs the QBE taking the QRI as query item. The remaining operation of search and customizing the results for mobile device screen are done in normal. The results of 'Find Similar' feature are shown in Figure 19.

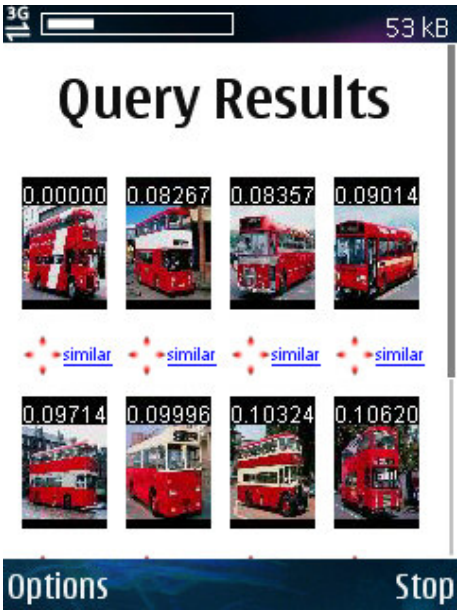

Figure 19. Results of 'Find Similar' feature

We can also download the full sized image in its' original format by clicking on any of the QRI, JspMuvis calls it retrieved media, it is shown in figure 20.

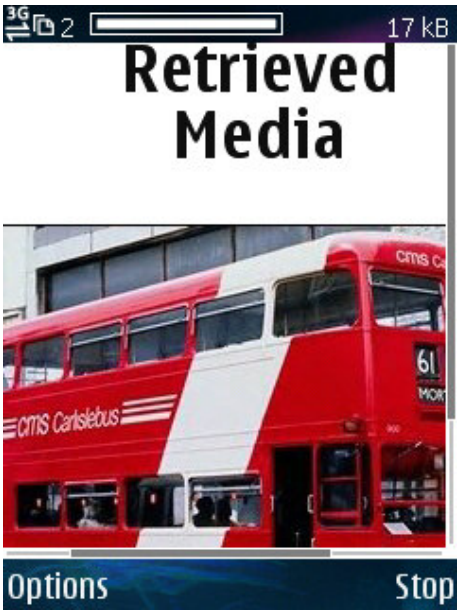

Figure 20. QRI downloaded on mobile device

Nokia N81 has display of 320x240 pixels and Figure 21 shows different screens of JspMuvis. Figure 21(A) shows the first screen of JspMuvis on Nokia N81. We chose to use the "Random Query" feature for this phone where a random image is taken from the database and searched among all the images present in multimedia database. The resultant images which are formatted and customized in size for Nokia N81 are shown in Figure 21 (B). Figure 21 (C) shows the first resultant image shown in its original size.

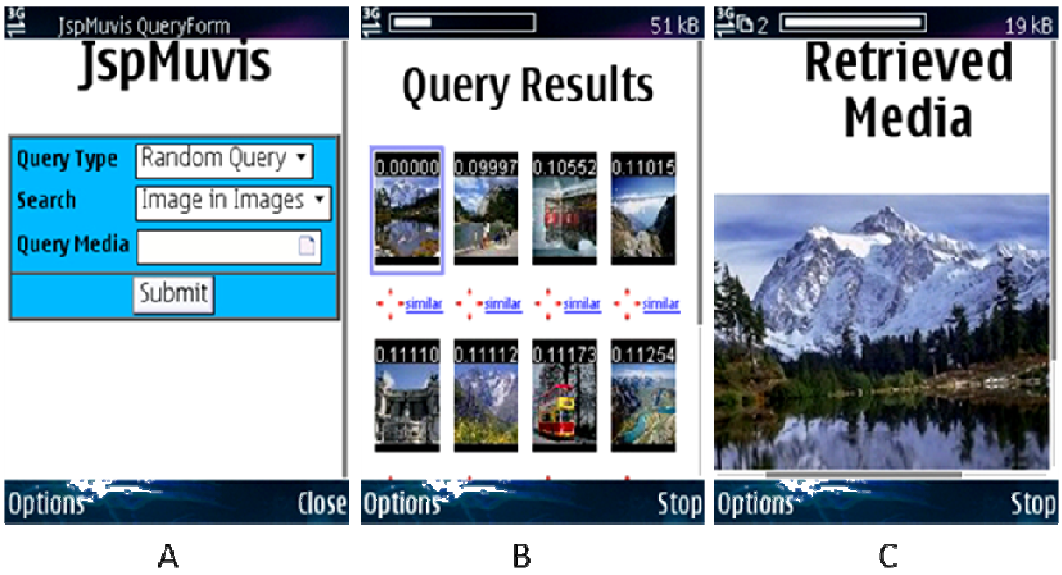

Figure 21. Random Query performed on Nokia N81

Above experiments show that the images resulted from the content based query are formatted and resized according to Nokia N95 and Nokia N81. It is quite evident

from Figure 18 and Figure 21 B that the resultant images are grouped properly and they fit perfectly on the mobile device's screen. We can expect the similar results for all the mobile devices that have their User Agent profiles available.

8.2. Performance Tests using different Network Usage

The client Query time and the server query time is noted and analysis chart is produced. The sample database used in our experiments contains 476 images in different formats. JspMuvis server is activated on a PC equipped with P4 2.99GHz and 1.9GB of RAM. Basic visual features such as YUV, HSV, RGB color histograms, Gray Level Co-Occurrence Matrix (GLCM) as a texture feature [46] are extracted for the sample image database operated by JspMuvis server. Since we are measuring performance of JspMuvis over different networks we do not have to use several mobile devices. We chose Nokia N9500 which supports 3G, EDGE and WLAN network technologies.

Connection Type	**CQT**		**SQT**	
	Mean (ms)	SD (ms)	Mean (ms)	SD (ms)
3G	7100	567	2118	11
EDGE	7628	479	2118	7
WLAN	3462	158	2122	6

Table 1: Mean and SD of CQT and SQT for different network

A content-based query operation is initiated from a client device to retrieve the similar images with respect to a query image. Server Query Time (*SQT*) is measured as the time spent to perform a query operation on the server side whereas Client Query Time (*CQT*) is the entire time passed from sending a query request, performing the query on the server side, formation of the *QRI* on server side and the to the reception of *QRI* to the client. Statistics (mean and Standard Deviation (*SD)*) of *CQT* and *SQT* are presented in Table 1. Each mean and SD is calculated over 12 query operations. *CQT* is significantly high as shown in Table 1 but a direct comparison with SQT approves that this is due to the network conditions. The advances in mobile network technologies in the near future might close the gap between CQT and SQT. Furthermore, some reduction methods on network traffic such as session tracking [25] can be used to minimize the flow of data between mobile device and the server.

As stated earlier *QRI* as a JPEG (Joint Photographic Experts Group) image is created on the server side to embed the query results and display all in once in the browser GUI on a mobile device. Since today's mobile devices usually do not support high-resolution screens; a low quality factor in JPEG format can be used conveniently for the *QRI* creation in order to reduce the *QRI* size and thus in effect to reduce the

transmission period. During our experiments we observed that JPEG quality factor 0.20 to 0.25 is enough to display the *QRI* without a significant degradation in visual perception for most of the mobile devices.

9. Conclusion

The research question presented and addressed in this thesis work was related to displaying the images to mobile device screens in a customized and optimal way.

A novel CBIVR framework for the connected mobile devices is presented for the efficient content management of personal multimedia archives from connected mobile devices. JspMuvis is designed as a web application, which makes it easier to be used across a range of mobile devices. The reduction of network bandwidth usage and the consumption of processor resources are two of the major factors in the success of mobile device applications. JspMuvis reduces the latter via performing the operations requiring a significant CPU power and memory on the server side, which is activated on a powerful PC. With the help of session tracking we further reduced the excessive network traffic between client and server. JspMuvis has the capability of receiving the mobile device capability information on server side, which is running on powerful computer, and with the implemented algorithms on server side it is possible to customize/resize the query resulted content according to mobile device specifications. Therefore when the result of content based search reaches mobile device, it is presentable on desired screen. So with less network bandwidth usage JspMuvis is able to display images and videos on any mobile device's screen in an optimal way.

By using metric access method based indexing algorithms particularly designed for multimedia databases on the server side, CQT can be reduced especially for large multimedia databases. Currently, JspMuvis server acts only as a CBIVR server. The scalability of JspMuvis allows it to be upgraded to a fully-functioning content-based multimedia indexing and retrieval server. It is expected that in future there will me more server applications developed that generate multimedia content for mobile devices, the features of JspMuvis such as, device capability capturing and content customization, can be separated and used as plug-in or filters for those applications.

10. Definition of Terms, Acronyms and Abbreviations

This section provides the definitions of all terms, acronyms, and abbreviations required to interpret them properly.

Term	Description
2D	2 Dimensional
AFeX	Audio Feature Extraction
API	Application Programming Interface
AV	Audio-Visual
AVI	Audio Video Interlaced (Microsoft ©)
Browser	An agent that allows a user to perceive and interact with information on the Web.
CBIR	Content-based Image Retrieval
CC/PP	Composite Capabilities/Preferences Profile
Client	The role adopted by an application (running in browser) when it retrieves and/or renders information from server.
CPU	Central Processing Unit
Delivery Context	A set of attributes that characterizes the capabilities of the access mechanism, the preferences of the user and other aspects of the context into which a web page is to be delivered
FeX	Feature Extraction
Gateway	A gateway is an intermediary which acts as a server on behalf of some other server with the purpose of supplying resources or resource manifestations from that other server. Clients using a gateway know the gateway is present but do not know that it is an intermediary.
GUI	Graphical User Interface
HTTP Client	A program that establishes connections for the purpose of sending HTTP requests
HTTP Request	An HTTP message sent by an HTTP client requesting that some operation be performed on some resource. Also, the act of sending such a message is termed making a request.
HTTP Response	An HTTP message sent back to an HTTP client in response to a previous HTTP request.
HTTP Server	An application program that accepts connections in order to service HTTP requests by sending back HTTP responses.
ISO	International Organization for Standardization
JPEG	Joint Pictures Expert Group
JSP	Java Server Pages
M-MUVIS	Mobile Multimedia Video Indexing and Retrieval
MUVIS	Multimedia Video Indexing and Retrieval
NQ	Normal Query
OS	Operating System
PQ	Progressive Query
QBE	Query by Example
QP	Query Path
QTT	Query Total Time

TUT	Tampere University of Technology
UAProf	User Agent Profile
UI	User Interface
User Agent	A client within a device that performs rendering. When a browser makes the request to server, it sends a text string to identify the user agent.

References

[1] J. Lempiäinen, M. Manninen, Radio Interface System Planning for GSM/GPRS/UMTS, Published by Kluwer Academic.

[2] Douglas Boling, Programming Windows CE, 3rd Edition, by Microsoft

[3] “Symbian OS”, http://www.symbian.com

[4] “Opera” http://www.opera.com/products/mobile/operamini/

[5] “Java” http://java.sun.com/

[6] "HTML 4" http://www.w3.org/TR/html4/

[7] "XML" http://www.w3.org/TR/xml/

[8] “W3C” http://www.w3.org/

[9] “SVG” http://www.w3.org/TR/2006/CR-SVGMobile12-20060810/

[10] Clinton Wong, HTTP Pocket Reference, published by O’Reilly.

[11] "User Agent" http://en.wikipedia.org/wiki/User_Agent

[12] "UAPROF2" http://www.w3.org/TR/CCPP-struct-vocab2/#ref9

[13] "OMA" http://www.openmobilealliance.org/

[14] Klyne, G. et al, 'Composite Capability / Preference Profiles (CC / PP): Structure and Vocabularies 1.0', World Wide Web Consortium, 15 Jan 2004.

[15] “SAP” http://www.sap.com/index.epx

[16] “Google Mobile Email” http://www.google.com/mobile/mail/

[17] “BBC Mobile News” http://www.bbc.co.uk/mobile/

[18] Serkan Kiranyaz, "Advanced Techniques for Content-Based Management of Multimedia Databases", PhD. Thesis at Tampere University of Technology, Tampere, Finland, June 2005

[19] “MUVIS” http://muvis.cs.tut.fi

[20] J.R. Smith and Chang, “Visual SEEk: A fully automated content-based image query system”, ACM Multimedia, Boston, Nov. 1996.

[21] A. Pentland, R.W. Picard, S. Sclaroff, “Photobook: tools for content based manipulation of image databases”, Proc SPIE (Storage and Retrieval for Image and Video Databases II) 2185:34-37, 1994.

[22] “Virage”, http://:www.virage.com

[23] O. Guldogan, M. Gabbouj, “Content-based image indexing and retrieval framework on symbian based mobile platform”, European Signal Processing Conference, EUSIPCO 2005, Antalya, Turkey, Sep. 2005.

[24] I. Ahmad, Moncef Gabbouj, “Compression and Network Effect on Content-Based Image Retrieval on Java Enabled Mobile Devices” FINSIG’05, pp35-38, University of Kuopio, Finland, August 2005.

[25] Sing Li, Paul Houle, Mark Wilcox, Ron Phillips, Piroz Mohseni, Stefan Zeiger, Hans Bergsten, Matthew Ferris, Danny Ayers, “Professional Java Server Programming”, published by Peer Information Inc., ISBN: 1861002777.

[26] “TomCat” http://jakarta.apache.org/tomcat/

[27] ”Midlet” http://java.sun.com/j2me

[28] R. Sarvas, E.Herrarte, A.Wilhelm, and M.Davis, “Metadata Creation System for Mobile Images”, Proc. of the 2nd international conference on Mobile systems, applications, and services, MobiSys, Boston USA, Pages: 36 -48, June 2004.

[29] K.Tollmar, T.Yeh and T.Darrell, “IDeixis: image-based Deixis for finding location-based information”, Mobile HCI, Vienna, Austria, Pages: 781 – 782, 2004.

[30] “Video Formats” http://en.wikipedia.org/wiki/YUV

[31] J. Keogh, “The Complete Reference J2ME”, published by Osborne/McGraw-Hill, February 27, 2003.

[32] “Nokia”, http://www.nokia.com/

[33] I. Ahmad, F. Alaya Cheikh, B. Cramariuc and M. Gabbouj, “Query by Image Content using NOKIA 9210 Communicator”, Proc. of the Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS'01, pp.133-137, Tampere, Finland, May 2001.

[34] Sing Li, Paul Houle, Mark Wilcox, Ron Phillips, Piroz Mohseni, Stefan Zeiger, Hans Bergsten, Matthew Ferris, Danny Ayers, “Professional Java Server Programming”, published by Peer Information Inc., ISBN: 1861002777.

[35] “RDF" W3C recommendation” http://www.w3.org/TR/rdf-primer/

[36] “XML Namespaces” http://www.w3.org/TR/REC-xml-names/

[37] “RDF Schema” http://www.w3.org/TR/1999/PR-rdf-schema-19990303/

[38] WAG UAProf', Wireless Application Protocol Forum, 20 Oct 2001.

[39] "Deli" http://sourceforge.net/projects/delicon/

[40] "ARP" http://www.hpl.hp.com/personal/jjc/arp/

[41] "Jena" http://jena.sourceforge.net/

[42] "RDF validator" http://www.w3.org/RDF/Validator/

[43] http://www.w3.org/TR/rdf-syntax-grammar/

[44] "XHTML" http://www.w3.org/TR/xhtml1/

[45] “DOM” http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/

[46] M. Partio, B. Cramariuc, M. Gabbouj, A. Visa, “Rock Texture Retrieval Using Gray Level Co-occurrence Matrix”, In Proc. of 5th Nordic Signal Processing Symposium, October 2002.

Appendices

A. [Scalable Vector Graphics (SVG)]

http://www.w3.org/TR/2006/CR-SVGMobile12-20060810/

B. [.NET Mobile Images]

Source: http://www.w3schools.com/dotnetmobile/mobile_images.asp

.NET Mobile displays different types of images for different types of devices.

The Image Control allows the developer to specify different types of images for different types of devices. Some mobile devices will display GIF images. Other mobile devices will display BMP images or WBMP images. The Image Control allows you to specify different images for each preferred image type.

This mobile page displays an image:

```
<%@ Page
Inherits=
"System.Web.UI.MobileControls.MobilePage"%>
<%@ Register
TagPrefix="Mobile"
Namespace="System.Web.UI.MobileControls"
Assembly="System.Web.Mobile" %>

<Mobile:Form runat="server">
<Mobile:Image runat="server">
  <DeviceSpecific>
    <Choice ImageURL="image.gif" />
    <Choice ImageURL="image.bmp" />
    <Choice ImageURL="image.wbmp" />
  </DeviceSpecific>
</Mobile:Image>
</Mobile:Form>
```

When this page is displayed on pocket PC, a GIF image will be displayed. On a cell phone a WBMP image or a BMP image will be displayed, according to the characteristics of the cell phone.

C. [Nokia E65 UAProfile]

http://nds.nokia.com/uaprof/NE65-1r100.xml

D. [UAProf profiles]

http://delicon.sourceforge.net/profiles.html

E. [JSR 188: CC/PP Processing]

http://www.jcp.org/en/jsr/detail?id=188

F. [CC/PP Profile]

Source: http://www.w3.org/TR/2003/PR-CCPP-struct-vocab-20031015/#CCPPArchitecture

The initial branches of the CC/PP profile tree describe major components of the client. Examples of major components are:

- The hardware platform upon which software is executing,
- The software platform upon which all applications are hosted, or
- An individual application, such as a browser.

A simple, graphical representation of the bottom of a CC/PP tree based on three components (`TerminalHardware`, `TerminalSoftware` and `TerminalBrowser`) would be:

[example:MyProfile]
|
+--ccpp:component-->[example:TerminalHardware]
+--ccpp:component-->[example:TerminalSoftware]
+--ccpp:component-->[example:TerminalBrowser]

The corresponding XML might look like this:

Figure 2-1b: CC/PP profile components in XML

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ccpp="http://www.w3.org/2002/11/08-ccpp-schema#"
         xmlns:example="http://www.example.com/schema#">

  <rdf:Description rdf:about="http://www.example.com/profile#MyProfile">

    <ccpp:component>
      <rdf:Description
        rdf:about="http://www.example.com/profile#TerminalHardware">
       <!--  TerminalHardware properties here  -->
      </rdf:Description>
    </ccpp:component>
```

```
    <ccpp:component>
      <rdf:Description
          rdf:about="http://www.example.com/profile#TerminalSoftware">
        <!--  TerminalSoftware properties here  -->
      </rdf:Description>
    </ccpp:component>

    <ccpp:component>
      <rdf:Description
          rdf:about="http://www.example.com/profile#TerminalBrowser">
        <!--  TerminalBrowser properties here  -->
      </rdf:Description>
    </ccpp:component>

  </rdf:Description>
</rdf:RDF>
```