

---

# Audio-Visual Multimedia Retrieval on Mobile Devices

Iftikhar Ahmad<sup>1</sup> and Moncef Gabbouj<sup>2</sup>

<sup>1</sup> Nokia Corporation, P.O. Box 88, FIN-33721, Tampere, Finland  
iftikhar.ahmad@nokia.com

<sup>2</sup> Institute of Signal Processing, Tampere University of Technology, P.O. Box 553,  
FIN-33101, Tampere 33720, Finland  
moncef.gabbouj@tut.fi

**Summary.** Rapidly increasing digital media technologies and compression standards combined with today's multimedia mobile devices and the Internet have led to the widespread exchange of multimedia contents. This, however, brings the problem of handling and accessibility of such a massive digital media from a variety of devices (hand held phones to personal computers). Mobile devices are not only limited in size, shape and input/output capabilities, but also have limited processing power, storage space, battery power and proprietary application programming interfaces. Therefore content-based multimedia retrieval from mobile devices is a challenge.

## 8.1 Introduction

The amount of personal digital information is increasing at an enormous rate. New multimedia mobile devices with integrated microphones and cameras facilitate the creation of audiovisual content. These mobile devices are no longer used for voice communication only. Nowadays, they are more frequently used to capture and manipulate different media types and run different applications on the devices. Additionally, when combined with Wireless Local Area Network (WLAN) and 3G network technologies, they may provide high speed access to the wealth of media items on the Internet. Therefore, accessing multimedia items from these mobile devices is no longer a problem. However, retrieving a specific media item of interest from a multimedia database using a mobile device remains a challenging research problem. In this context, one particular user scenario might be the following: Using a multimedia mobile device, a user can capture an image or record an audio/video clip and run an application to perform a content-based query-by-example (QBE) operation virtually from anywhere. However, Content-Based Multimedia Retrieval (CBMR) from mobile devices adds new challenges besides those encountered in typical CBMR operations on desktop machines. For instance, different



**Fig. 8.1.** Examples of different multimedia mobile devices

mobile devices come in different designs and capabilities. Moreover, they have different operating systems, input/output limitations and support different media file formats. Few examples of multimedia mobile devices from different vendor are shown in Fig. 8.1.

Recently, the capabilities of mobile devices have been significantly improved (faster input/output, bigger memory capacity, processing and battery power) but they are still far behind when compared to desktop computers. As a result, it is hard to provide a generic solution appropriate for all mobile devices and, therefore, special care must be taken when developing applications for them. To address the problem in wide range of devices, multimedia content presentation should be independent of mobile device hardware (user interface) and software (operating system). An efficient user interface is required to handle different media formats on a range of devices with diverse capabilities. There are various ways to achieve that by using interpreter languages such as Java [1], Python, etc. or by using markup languages [2] such as eXtensible Hyper Text Markup Language (XHTML) with device supported browsers.

Multimedia retrieval on mobile devices is an emerging research area [3–8]. With the existing mobile device operating systems such as Symbian OS [9], MS Windows Mobile [10], Linux [11], etc., it became possible to develop applications that can run on mobile devices to perform sophisticated media manipulation tasks. These efforts paved the way for the development of CBMR via mobile devices. Usually there are two approaches, one is on-device CBMR system and the other is distributed (client-server) CBMR system. In on-device CBMR, multimedia content and extracted features are saved on the device. In this context, most relevant efforts are the following: Gandhi et al. [7] proposed an intelligent multimedia content management on mobile devices with the help of metadata. Davis and Sarvas [5] developed a system for image retrieval over mobile devices using (textural) metadata associated with the images. In the system they proposed, the XHTML pages contain a large amount of redundant data, which eventually increases the retrieval time.

Lifeblog [12], an application from Nokia [13], uses textural metadata for organization and synchronization of mobile device content (images, audio, video, SMS, MMS etc.) with the PC. Modern multimedia mobile devices can generate very useful textural metadata (date, time, global position, Bluetooth neighbor names, camera focus info, etc.) for multimedia organization and retrieval. Humans can recall the information by date, places or with (relatives, friends) names efficiently, but getting all the above mentioned textural metadata along with multimedia item is very difficult or even not possible. For example, the global positioning system might not be ready when recording multimedia items. Therefore, we are focusing mainly on content-based multimedia retrieval in this chapter. Gulgodan et al. [8] proposed *on device* content-based image indexing and retrieval framework for Symbian series 60 devices. This approach presents several limitations; for example: mobile devices have proprietary Application Programming Interfaces (APIs) for handling (i.e. accessing, processing, editing, streaming, etc.) multimedia items, which limit the applications using them to a certain set of devices or certain platforms (operating systems). Another limitation is its large consumption of the device system resources (battery and processing power, etc). Even though a standalone CBMR system can be implemented on mobile devices, with the increasing number of multimedia items it might take an unreasonable amount of time to perform a content-based retrieval operation. Furthermore, such system will eventually reduce the mobile device talk and stand by times.

To avoid the limitations of on device storage, processing and battery power as mentioned above, a second approach is used, which is based on a client-server architecture. In client-server based CBMR, part of the program (client) resides on the device while the remaining part of it (server) resides on some other machine (such as PC). Examples are Mobile-MULTimedia Video Indexing and retrieval System (M-MUVIS) [3] and Java Server Pages MULTimedia Video Indexing and retrieval System (JspMUVIS) [6]. M-MUVIS is designed as a client-server architecture in which the server runs on a computer and the client runs on a mobile device. The content-based query operation and media handling (format conversion, resizing, watermarking for copyright, streaming, etc.) are performed on the server side, whereas the lightweight operations, such as sending a query request and User Interface (UI) capabilities, are handled on the mobile device. MULTimedia Video Indexing and retrieval System (MUVIS) is developed at Tampere University of Technology (TUT) as a multimedia feature extraction, indexing and retrieval system for personal computers. More information about MUVIS can be found from [14]. M-MUVIS and JspMUVIS extend MUVIS for mobile platforms.

Since Java is an interpreter language and device agnostic, a client application written in Java supports a wide range of devices as compared to other interpreted languages such as Python. Secondly, 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) [15]. 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.

MIDP provide means to place commands [15] (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. 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, even though 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 8.2 shows the M-MUVIS client UI on Nokia 6630 [13].

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. Mobile devices can use a Web or WAP browser to display those XHTML pages. JspMUVIS server (TomCat [16] web server [17]) runs on a computer. As most of mobile device users are novice users and not computer programmers, their mobile devices are for their personal usage only. 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 there is no need to install

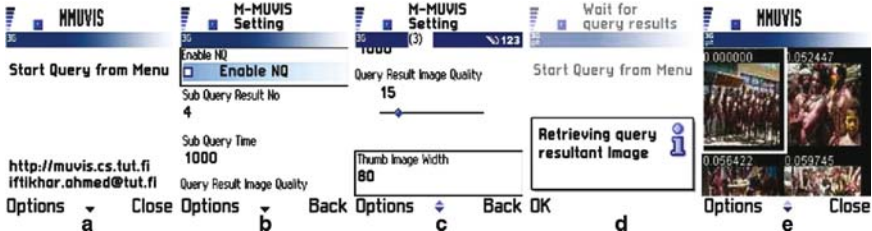


Fig. 8.2. M-MUVIS client user interface on Nokia 6630. (a) Main UI of M-MUVIS client (b) and (c) shows M-MUVIS setting view (d) Query result retrieval wait dialog is shown (e) Shows query resultant image

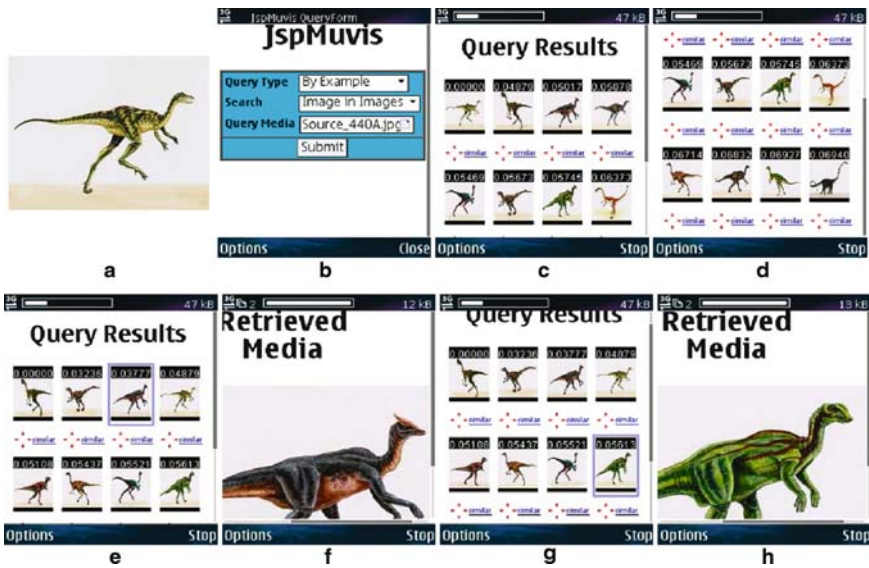


Fig. 8.3. JspMUVIS user interface on Nokia N90

any application on their devices, as XHTML pages are downloaded as in normal browsing and used for CBMR. Web or WAP browsers have no or just limited access to device resources such as camera, Bluetooth etc. JspMUVIS running on Nokia N90 [13] is shown in Fig. 8.3. Query Image is shown in Fig. 8.3a whereas JspMUVIS query form is shown in Fig. 8.3b. Figure 8.3c–h shows the query resultant image (i.e. similar images) and two selected images from the query result. In JspMUVIS, a lot of extra data is transferred between client and server due to XHTML tags that increase the query and media retrieval time.

The goal is to design and develop a content-based multimedia retrieval framework, which would enable any client supporting Java platform or XHTML browsing to retrieve multimedia items similar to a query media item from a multimedia database. Query operation and media retrieval (optionally conversion in different format) should be independent from each other so that one operation should not block the other operation. M-MUVIS server comprises two Java servlets [17] running inside a Tomcat web server which in effect performs the content-based query operation and media retrieval on the server side. Two servlets make the query and media retrieval two independent operations. The MUVIS Query Servlet (MQS) has native libraries for efficient multimedia query related operations. The second servlet, the so-called MUVIS Media Retrieval Servlet (MMRS), is used for the media retrieval. JspMUVIS uses similar approach of two separate Java beans [1] for query operation and media retrieval to make the query and media retrievals two independent operations.

## 8.2 Content-Based Multimedia Retrieval

Due to limited input capabilities and other limitations imposed by only text-based queries, CBMR promises a potential solution to the multimedia content management of mobile devices. CBMR addresses the problem of accessing a multimedia item that bears certain content and usually relies on the characterization of low-level features such as color, shape, texture, etc. Besides the content based query operation, content-based multimedia retrieval from mobile devices adds new challenges.

In the area of content-based media retrieval, most common query types are visual and audio. A visual query can be performed for still images and video clips; whereas, an audio query can be performed in order to retrieve similar audio and video (with audio) clips based on audio information. Video clips recorded on mobile devices usually contain both audio and video tracks, which form two distinct sources to extract visual and audio features for content-based query operation. In the literature, visual information has received more attention with regards to content-based information retrieval. However, in some cases audio can be even more promising than its visual counterpart since it is most likely to be unique and significantly stable within the entire duration of the content. Therefore, the M-MUVIS and JspMUVIS framework supports both audio and visual queries distinctively and as an alternative to each other whilst being performed over multimedia databases. In a query operation, similarity distances of multimedia items are calculated and ranking operations are performed afterwards; whereas, in a media retrieval operation “Query Resultant Image” (QRI) is created and streamed to the client. The first-best matched 12 images or key-frames of video clips are drawn as thumbnail images on QRI along with similarity scores. The first thumbnail image in the QRI is always the query image or the first key frame of the query video. This helps the user to quickly compare the query results. Example QRIs for Nokia 6630 and Nokia 9500 are shown in Figs. 8.4 and 8.5.

CBMR is now possible over mobile devices for several tens of thousands of image databases. Larger databases require additional effects, especially in the offline indexing side, in order to cluster similar items together for faster search and retrieval. Hierarchical Cellular Tree (HCT) is an attempt towards this challenge. More information about HCT can be found in [18].

### 8.2.1 Visual Query

Basic visual features such as YUV, HSV, RGB color histograms, Gray Level Co-Occurrence Matrix (GLCM) [19] as a texture feature, are used on the server side to perform content-based visual query operations. The most common retrieval scheme, known as normal query, is QBE. It constitutes the traditional query operation and is usually performed via an exhaustive search



Fig. 8.4. Visual query resultant image for Nokia 6630

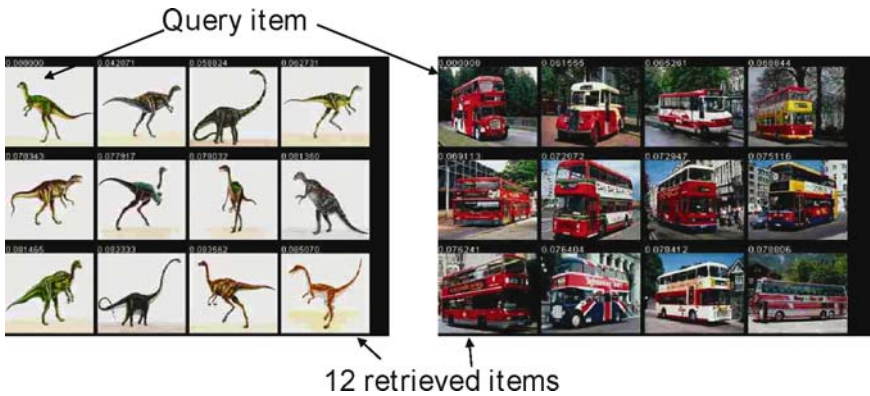


Fig. 8.5. Visual query resultant image for Nokia 9500

over the entire database due to lack of an efficient indexing scheme. In case of an indexed database, a query path is created in the database where most relevant items are placed first. Three visual QRIs created on M-MUVIS server for Nokia 6630 are shown in Fig. 8.4.

The sample database used in M-MUVIS and JspMUVIS experiments contains 1,000 images. The server runs on a PC equipped with Pentium 4, 2.99 GHz and 1.9 GB of RAM. Figure 8.5 shows two visual QRIs for Nokia 9500 created on the server side.

### 8.2.2 Audio Query

Audio information often helps to understand the content of digital media and, in certain cases, audio might even be the only source of information, as in audio-only clips for example. Therefore, audio information has been recently used for content-based multimedia indexing and retrieval [4, 20] and [21]. Audio may also provide significant advantages over the visual counterpart, especially if the audio content features extracted are similar to those used by the human auditory system. This, on the other hand, requires efficient and generic audio (content) analysis that yields robust semantic classification and segmentation. More details about audio classification and segmentation can be found in [22]. Most mobile devices provide support for audio recording. The user can record an audio clip and perform the content-based query operation. Usually, the audio content is smaller in size than the video content of the same duration, so they take less time during upload for the query operation.

The sample database used in M-MUVIS experiments contains 1,229 video clips. Each is of 1 min duration with a total duration of 20 hours. The clips are from the 1960s, but contain color video with audio. The features used in audio and visual queries performed in this section are as follows: Mel Frequency Cepstral Coefficients (MFCC) [22] for audio, YUV, HSV, RGB color histograms and the GLCM for the key frames of video. Figure 8.6 shows two QRIs one for visual and the other for audio query. The first key frame of the twelve best matched video clips is drawn on the QRI as thumbnail images. Audio-based video query generated QRI is shown in Fig. 8.6b. Similarly, Fig. 8.6a shows



Fig. 8.6. Retrieval results from a video database using: (a) visual content-based query and (b) audio content-based query

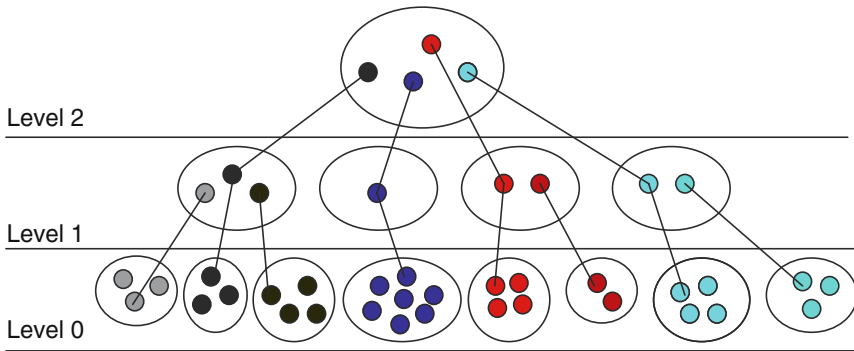


Fig. 8.7. An example of color-based indexed database

the visual query results. In this example, audio query results are better than the visual query, as the visual query has found only three similar clips whereas the audio query found twelve similar clips.

### 8.2.3 Guided tour of Multimedia Databases

In an indexed database, similar multimedia items are grouped together based on their features. An example is shown in Fig. 8.7 where circles (items) are grouped together based on their color feature in different cells. Each high level cell contains the representatives of low level similar color cells. The mobile device user can browse these cells up-down in different levels and left-right in each level in the selected database. HCT is an efficient indexing scheme developed in MUVIS and used in M-MUVIS and JspMUVIS. More information about HCT (i.e. item insertion, deletion or level creation etc.) can be found in [18].

The user can perform the query operation and from retrieved results he can select one media item of his interest and can view all the other members of the media item cell. The user can view next and previous cells or go up and down in the tree. This provides a guided tour to the multimedia items where user can see the similar items.

## 8.3 Client-Server Framework

Client-server framework handles the challenges of processing power and media storage for CBMR over mobile devices. A client is used to initiate the content-based query operation and sends it to the server, which in turn performs the query operation and sends the query results back to the client. With Client-server framework user can use personal computer or mobile devices to perform the CBMR to the same database by using the same feature set (audio

or video). It provides a uniform method to perform the query operation regardless of the machine (personal computer or mobile devices). Client-server framework raises the content privacy issues, as usually mobile devices contain personal contents. Mobile device users do want to share their multimedia contents with family members and friends but might not want to share it with others. Client-server framework can use authenticated and secure communication channel for media transfer and query operations.

### 8.3.1 Mobile-MUVIS Architecture

Client-server framework of M-MUVIS is shown in Fig. 8.8, where the client application is used to initiate the content-based query operation such as QBE 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 Fig. 8.8, there are two servlets (web applications) on the M-MUVIS server side: the MQS is used for performing content-based query operation, while the MMRS is used for the media retrieval operation. As shown in Fig. 8.8 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.

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

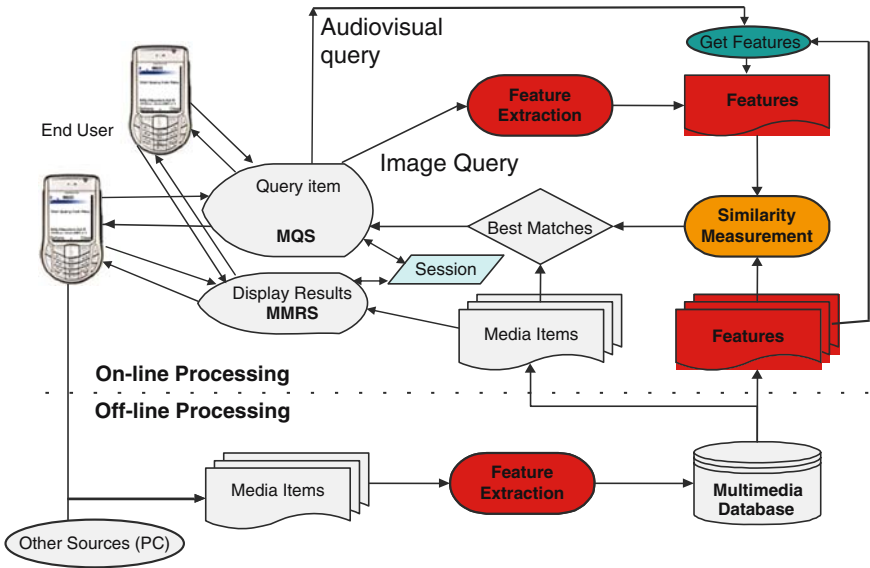


Fig. 8.8. M-MUVIS framework

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 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.

### 8.3.2 JSP MUVIS

XHTML is another choice to generate device independent content presentation. As most mobile devices support XHTML, it can be used for CBMR. Java Server Pages (JSP) can generate markup language XHTML pages; they are dynamically adaptable to different UIs of different devices with the help of on device browsers. JspMUVIS Server generates the HTML pages for PCs and XHTML pages for the mobile devices. As shown in Fig. 8.9, JspMUVIS Server creates and maintains a session [17] with the client. It also generates QRI for the client. JspMUVIS uses query engine to perform the query operation by using native (C/C++) library.

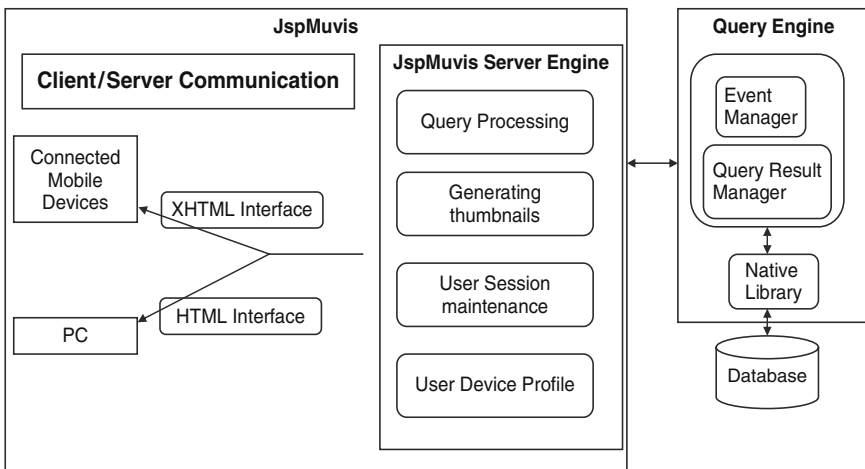


Fig. 8.9. JspMUVIS framework

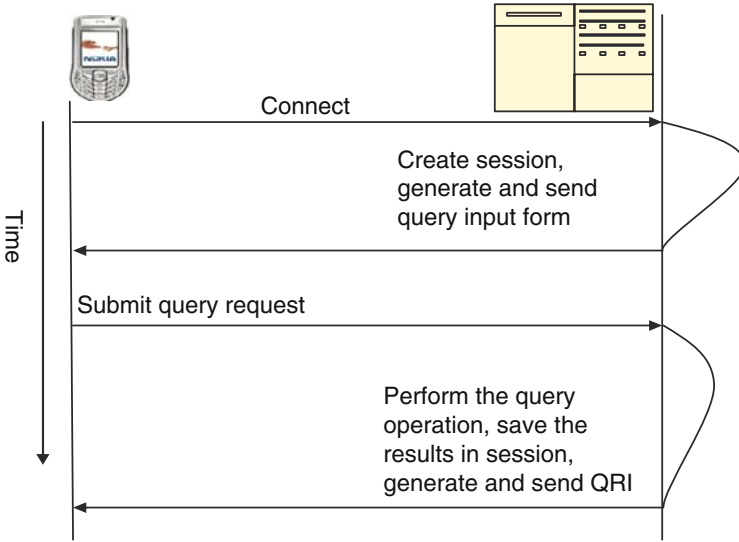


Fig. 8.10. JspMUVIS query operation

For optimum content generation for different clients, JspMUVIS server requires a description of the client capabilities. Two new compatible standards are defined for describing delivery context based on the resource description framework [23] composite capabilities/preferences profile [23] created by the W3C and user agent profile created by the WAP Forum [24]. These standards help in efficient delivery of context information (query information) to the server via low bandwidth wireless networks. JspMuvvis server receives the client request and gets the screen size, resolution and supported media types from the device profile [23]. With the help of deli library [23], JspMUVIS generates XHTML pages for mobile devices and HTML pages for PCs.

As shown in Fig. 8.10, first, the client connects to the server; the server creates a session and sends the query form to the client. The client sends the query request to the server and the server performs the query operation on the server side and generates the QRI according to the look and feel of the client. JspMUVIS operation is divided in three layers presentation, application and data layer. These layers are described below.

### 8.3.2.1 Presentation Layer

This is the user interface layer of JspMuvvis. XHTML pages generated by JspMUVIS server contain the user input form and the optional settings for a content-based query operation. Presentation layer is responsible for interacting with the user. It forms query operation and presents the query results.

### 8.3.2.2 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 takes 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. Application layer is responsible for session tracking of a particular JspMUVIS client. It uses cookies [17] to store the session information on the mobile device, and Internet browser picks the session information during the beginning of a client session.

### 8.3.2.3 Data Layer

Data layer is responsible for database operations. On the server side, native (C/C++) libraries are used for efficient content-based image query related operations. JspMuvvis contains Java beans, 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.

### 8.3.3 Interactive Query

Usually mobile device users cannot afford to wait for a long time to get the query results. In some networks (e.g. GSM) user is paying for air time. Alternatively an Interactive Query (IQ) allows the user to define the result retrieval time regardless of the database size. IQ is composed of a series of Progressive Sub-Queries (PSQs) where user can ask the results at any time during the query operation. IQ is designed to partition the database items into some sub-sets within which individual sub-queries can be performed. Therefore, a sub-query is a fractional query process that is performed over a sub-set of database items. Once a sub-query is completed over a particular sub-set, its results are merged with the last (overall) retrieval result to obtain a new (overall) retrieval result. This is a continuous operation, which proceeds incrementally, sub-set by sub-set to cover all the items within the database. The client can request for query results at any time during the query operation, IQ sends the results back to the client and also saves it to the client session on the server side. More information about IQ in MUVIS can be found from [25].

User can set the time delay for the query results on the client side. In this way query results delay can be deterministic regardless of the database size. Furthermore, when there is an efficient indexing structure available in the database, IQ can conveniently be used to retrieve the relevant items in the earliest possible time. For this purpose HCT indexing technique is used to perform similarity indexing in the M-MUVIS and JspMUVIS databases for efficient retrieval. An illustration of the IQ operation in M-MUVIS framework is shown in Fig. 8.11.

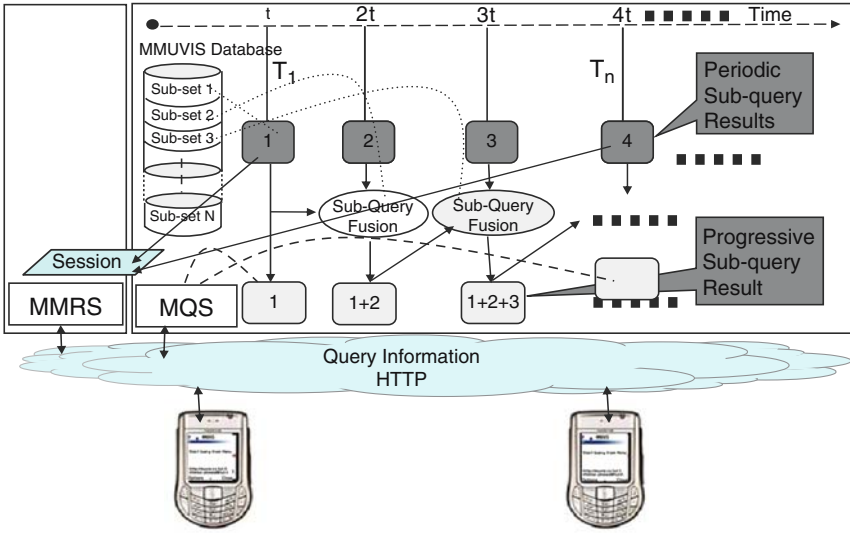


Fig. 8.11. Interactive query in M-MUVIS

As shown in Fig. 8.11 the client specifies time  $T_1$  for receiving the query results. IQ generates the query results after  $T_1$  time and streams it back to the client. The server continues the query operation till to the end of database and takes  $T_n$  time according to the database size. IQ helps to generate best matched media items in the early PSQs in an indexed multimedia database.

### 8.3.4 Network Communication

As mobile devices have limited network capabilities and it is the user who always bears the cost, the communication between the client-server is organized in such a way that all network demanding operations can be performed in a cheap or free networks (e.g. Bluetooth). Multimedia content transfer can be performed in cheap networks whereas query operations are performed online, in any network (GSM, GPRS, 3G and WLAN etc.). Usually mobile networks have less upload channel bandwidth than download. Communication should be designed in a way to limit the upload information and reduce information exchange between client and server. Therefore, in M-MUVIS and JspMUVIS, all device specific information is saved in a session on the server side. The client does not need to resend the device specific information, thus reducing the network traffic (specially upload). The client connects to the server, creates a session and during a session the client can perform many query operations and retrieve the query results. Mobile device users can share the query results by short message service (SMS), multimedia messaging service (MMS) or Bluetooth.

As Hyper Text Terminal Protocol (HTTP) [26] is supported by most devices, it is therefore used for communication between client and server. Since

HTTP is a stateless protocol, a session is created on the server side to store the configuration parameters sent with the query request by the M-MUVIS client and in case of JspMUVIS, the browser maintains the session. Each session has its own unique session identifier number. The client uses the session identifier for each transaction with the server. Additionally the query results are stored into the session. Therefore, the M-MUVIS and JspMUVIS client retrieve the query results from the session. In this way the URLs of the media items are not transmitted to the client explicitly in order to keep a high-level of security in the framework.

Table 8.1 presents the retrieval time statistics (mean ( $\mu$ ) and standard deviation ( $\sigma$ )) of ten audio query operations over a video database. The Server Query Time (SQT) is the time spent to perform a query operation on the server side; whereas, Client Query Time (CQT) is the entire time passed between sending a query request and performing the query operation on the server side. The media retrieval and media formatting are not included in both CQT and SQT measurements.

Table 8.2 presents the retrieval time statistics (mean ( $\mu$ ) and standard deviation ( $\sigma$ )) of ten visual query operations over different image databases.

**Table 8.1.** Audio query time statistics in video database

Network	IQ				NQ			
	CQT (ms)		SQT (ms)		CQT (ms)		SQT (ms)	
	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$
Audio query in video database								
6630 (3G)	6,777	949	5,578	33	297,385	3,955	290,465	344
9500 (WLAN)	5,609	104	5,005	8	290,281	352	289,654	306

**Table 8.2.** Visual query time statistics in image databases

Network	IQ				NQ
	CQT (ms)		SQT (ms)		CQT (ms)
	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$
1,000 image database					
6630 (3G)	1,329	1,581	113	6	1,401
9500 (Edge)	3,077	435	92	8	3,101
9500 (WLAN)	667	73	86	9	690
10,000 image database					
6630 (3G)	2,687	816	1,792	41	9,797
9500 (Edge)	6,417	1,010	2,102	37	9,828
9500 (WLAN)	2,720	452	2,091	50	6,515
60,000 image database					
6630 (3G)	9,421	640	5,160	10	411,094
9500 (Edge)	9,931	602	5,285	8	422,365
9500 (WLAN)	5,831	590	4,932	7	415,032

Nokia 6630 mobile phone can use 3G networks in a query operation whereas Nokia 9500 can use EDGE and WLAN technologies.

$$CQT = QIFET + SQT + CSCT \tag{8.1}$$

Query Image Feature Extraction Time (QIFET) is negligible when compared to Server Query Time (SQT) and Client Server Communication Time (CSCT). According to (1) a major portion of CQT is used in CSCT. CSCT is directly proportional to the amount of data transfer between the client and server. CSCT is reduced considerably by reducing the data exchange between the client and server specially reducing upload. In such a case, session tracking is introduced on the server side. All the client and query related information is saved in the session, and later that information can be used between the M-MUVIS client and server (CQS and MMRS). As stated before, mobile devices have limitations on the upload and download channels. In M-MUVIS and JspMUVIS framework we have focused on reducing the data exchange (especially upload information) with the help of session tracking. The client gets the intermediate query results and later can retrieve the complete query results.

### 8.4 User Interface and Multimedia Content Adaptation

The inherent small size and low weight of mobile devices imposes certain constraints on their hardware and user interface capabilities. Additionally mobile devices have different screen sizes and resolutions. These constraints create challenges for the effective access and presentation of information on such devices. M-MUVIS client active on Nokia 6630 is shown in Fig. 8.12.

To overcome these challenges, UI display capability of the client should support different screens sizes and resolutions. Therefore, during startup phase of M-MUVIS client, UI is adapted according to the device screen size. The same M-MUVIS client application can thus be used on different devices with different capabilities such as different screen sizes and supported contents.

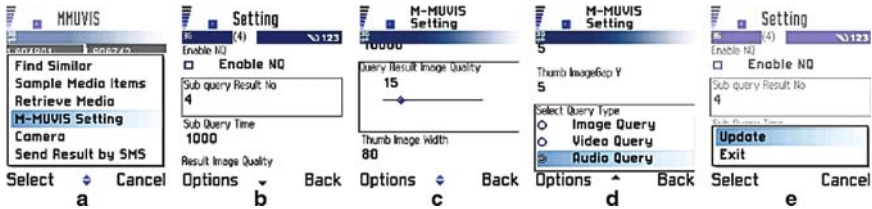
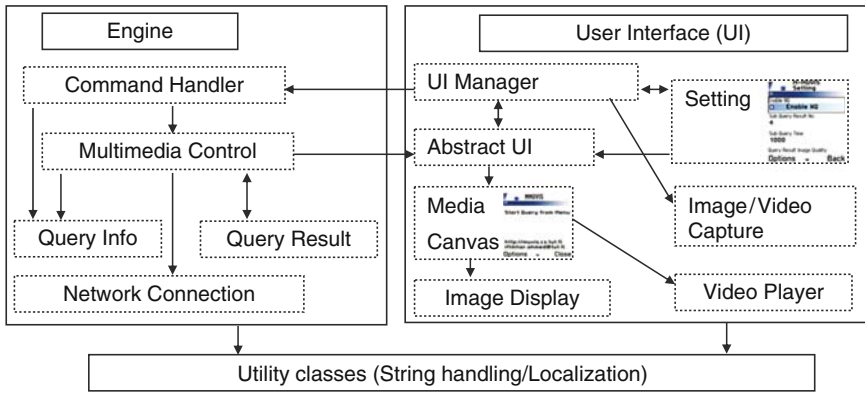


Fig. 8.12. Screen shots of M-MUVIS query configuration: (a) Main menu, (b) The query settings, (c) JPG image quality for the QRI, (d) Query type selection, (e) Updating the settings



**Fig. 8.13.** Block diagram of M-MUVIS client on mobile devices

As shown in Fig. 8.13, M-MUVIS client application consists of three parts. UI, Engine and Utility to cope with the different size and shape of the mobile devices, the UI module is further divided into two parts. An abstract layer and UI Manager are interacting with the Engine. The Engine then uses Abstract UI to update the UI. The UI manager combines with the image canvas and image capture to provide a layer which adapts to the different shapes and sizes of mobile devices. It is also responsible for command placements, as different devices have different keys for the same command operation. Command placement will help give the user the same look and feel as the native (device) applications. Engine receives commands from the UI to perform the required operation and updates the UI accordingly. The user can initiate several operations such as content-based query operation, change of M-MUVIS client settings, etc. and Engine uses the UI to display the Query Resultant Image (QRI) or images.

Engine is responsible for the activities behind the UI. It determines the thumbnail size according to the screen size of the device during the start of the M-MUVIS client application. The user may then change the default setting of the thumbnail image size from the settings dialog. When the user initiates a content-based query, the engine performs the query operation by contacting MQS. After the query operation, it sends a QRI retrieval request to the MMRS. Engine is also responsible for maintaining the list of thumbnails. It can later be used to retrieve a particular image or to initiate another query using the menu. The M-MUVIS client's view and query operation, with retrieved results, on Nokia N93, Nokia 6630 and Nokia 9500 are shown in Figs. 8.14–8.16 respectively.

Utility part is responsible for the string handling and localization of M-MUVIS client application. As mobile devices are for the personal use of general public usually the software of mobile devices should be localized in different languages to make the device usable for every one. Utility part in M-MUVIS client application is responsible for localization.

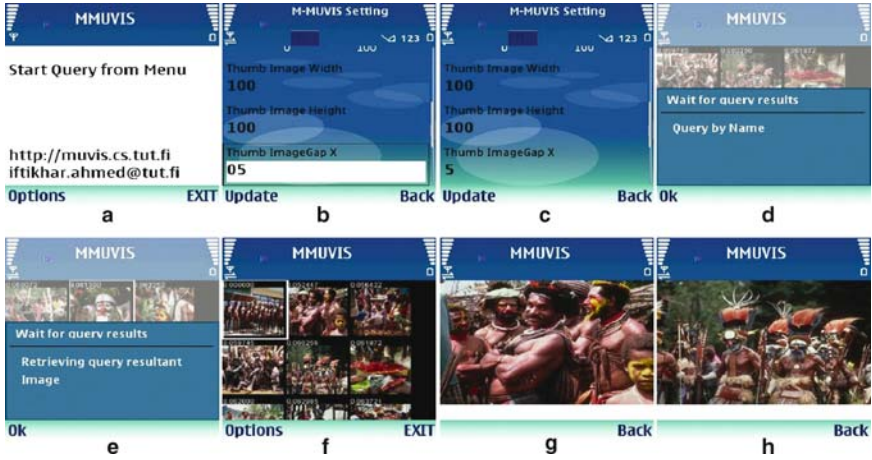


Fig. 8.14. M-MUVIS client on Nokia N93: (a) Main view (b) and (c) setting dialog, (d)–(h) show operation and media retrieval from image database

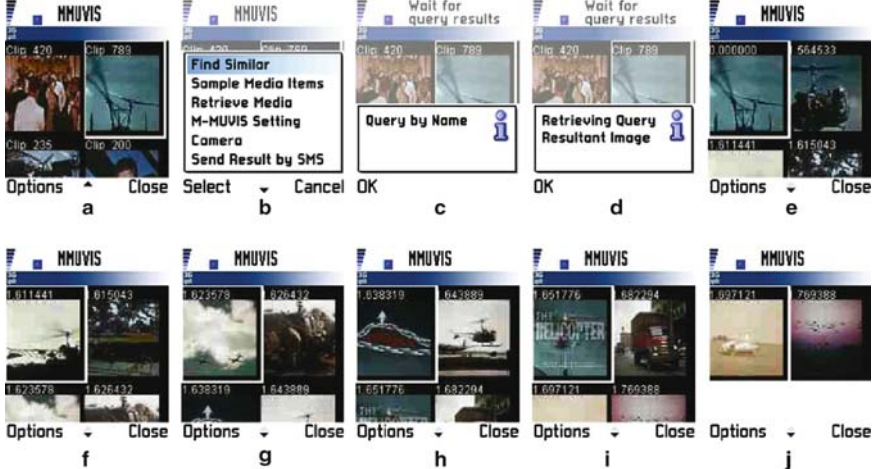


Fig. 8.15. Screenshot from a query operation in M-MUVIS: (a) Key frames of randomly selected clips from a database residing on the server side. (b) Query operations menu. (c) An ongoing query operation. (d) Retrieving query resultant image. (e)–(j) Key frames of the best matched video clips

M-MUVIS client performing the visual query operation in image database is shown in Fig. 8.17.

JspMuvvis on Nokia N95 is shown in Fig. 8.18. In Fig. 8.18a, JspMUVIS URL in the device browser is shown, a query form is shown in Fig. 8.18b where the user can select the query type, database (i.e. image, video or audiovisual) and media item from mobile device. Image item shown in Fig. 8.18g is selected in Fig. 8.18c–f.

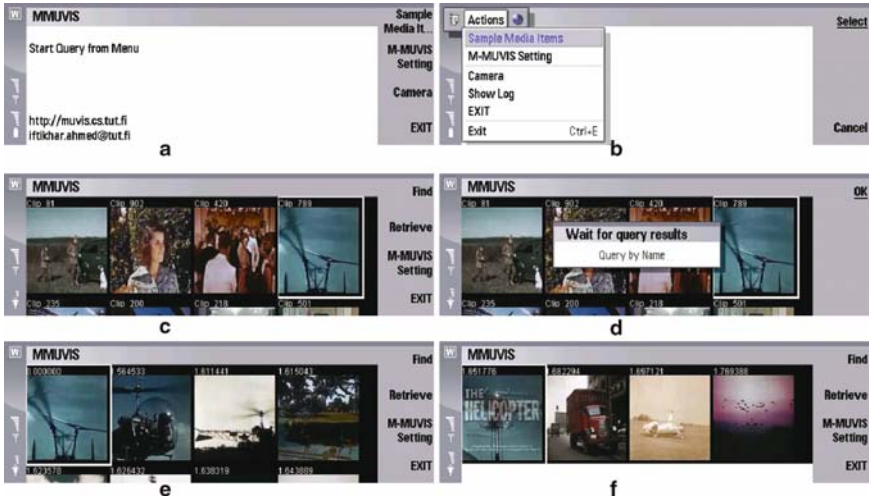


Fig. 8.16. Audio-based video retrieval operation from M-MUVIS client application on 9500: (a) Main view of M-MUVIS client (b) Command Menu (c) Key frames of randomly selected clips from a database residing on the server side. (d) Query operation. (e) and (f) Key frames

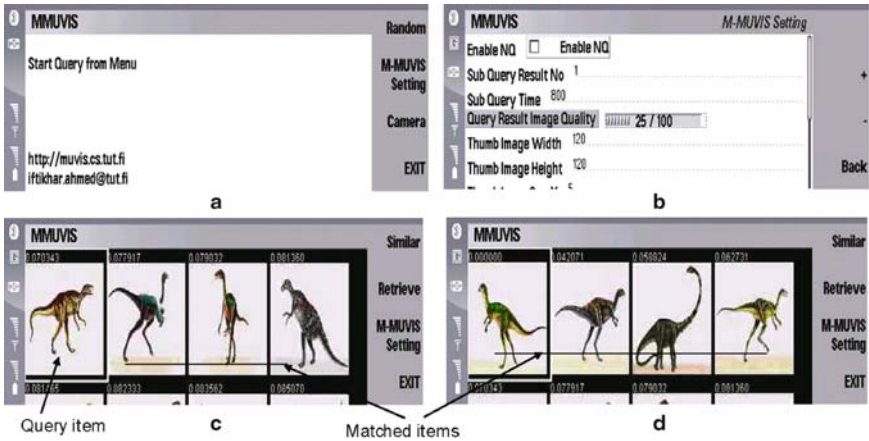


Fig. 8.17. Screen shots of M-MUVIS client on Nokia 9500: (a) Main view (b) Setting dialog (c) and (d) Visual query results are shown

In Fig. 8.19a–h the query results and three selected images on Nokia N95 are shown. The features used in visual queries performed in JspMUVIS are as follows: YUV, HSV, RGB color histograms and the GLCM for the images.

Figure 8.20 shows query image along with similar items on Nokia N95. Figures 8.18–8.20 show the JspMUVIS query operation on Nokia 95.



Fig. 8.18. Screenshot of JspMUVIS on Nokia N90 is shown

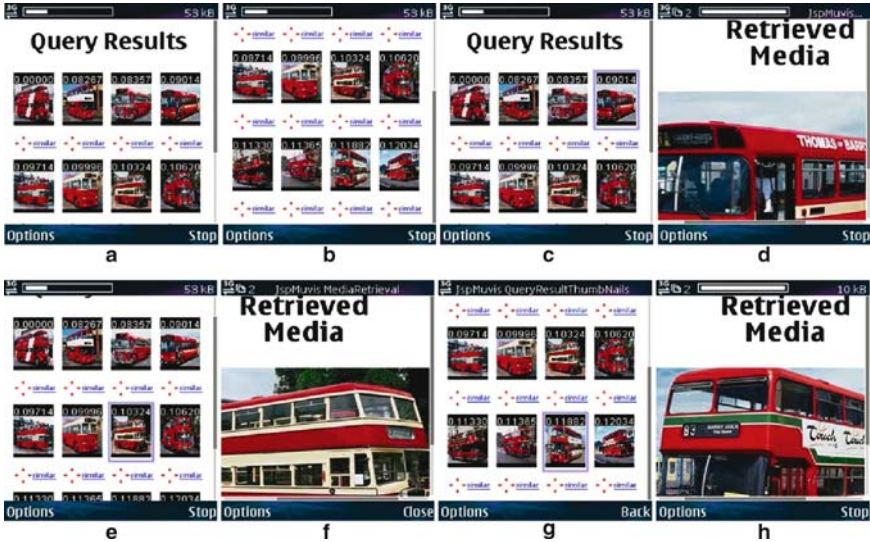
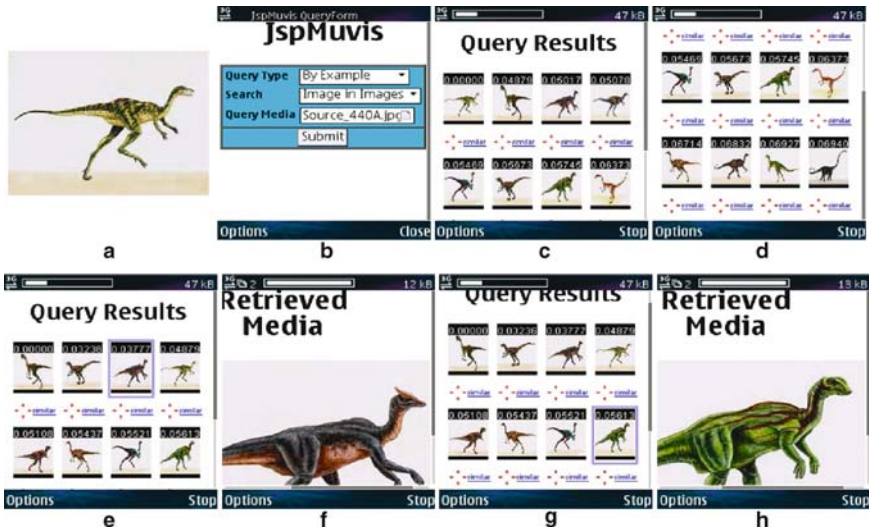


Fig. 8.19. JspMUVIS image query result on Nokia N95, (a)–(h) shows the query result and three selected image



**Fig. 8.20.** JspMUVIS query from Nokia N95, (a) Shows the query image on the device, (b) Shows the query form, (c)–(h) shows the similar images of the query item and two selected images

#### 8.4.1 Multimedia Content Adaptation for Mobile Devices

As mobile devices are not only in different shapes and sizes, different devices support different multimedia formats. The client application running on the mobile device can provide information about the device capabilities (supported media formats, screen size etc.) to the server. Then the server can provide client specific results such as image size according to the device screen size, highly compressed images for low resolution and limited bandwidth devices. Server can also provide device specific content in a specific format (JPG, PNG, H263, H264 etc.) and optionally can add watermarking for copyrights. JspMUVIS uses deli library to get the device profile and save the device specific information in the session. Later that information is used for efficient media retrieval. As most of the mobile devices are not having high resolution screens so a high compression can be used conveniently. A study in M-MUVIS shows that JPG quality factor 20 can be used conveniently in creating the QRI for mobile devices without any noticeable degradation in user experience.

In client-server framework, server can convert a media type to the device supported media format on the fly but that loads the server. As few multimedia formats are more popular than others, all multimedia items are converted offline to one format such as JPG for images, 3GPP for video and AMR for audio. These media formats can be further converted to other formats on the fly if request.

## 8.5 Conclusion

Client-server framework provides a solution for CBMR from mobile devices where client adapts to the look and feel of the device and provides information about the device. The client, which runs on a mobile device, can perform audiovisual-based queries within multimedia databases located on a remote server. The server performs the query operation and generates device specific content adaptable for the device. IQ over indexed database provides faster retrieval of most relevant media items in early interactive subquery. Communication between client and server should be optimized for efficient information exchange.

The ultimate goal for content-based multimedia management from mobile devices is that, the user can record his/her query (audio, audiovisual) and then possibly speech to text conversion is performed on the recorded multimedia if needed. The generated text is used as hint for metadata selection and for CBMR operation on the server side. Query results are generated and resultant media items are converted to the device supported format and converted results are streamed to the client as retrieval results.

## References

1. Java, <http://java.sun.com/>
2. XHTML, <http://www.w3.org/MarkUp/>
3. I. Ahmad, S. Kiranyaz and M. Gabbouj, An Efficient Image Retrieval Scheme on Java Enabled Mobile Devices, MMSP 05, International Workshop on Multimedia Signal Processing, Shanghai, China, 30 Nov. 2, 2005.
4. I. Ahmad, S. Kiranyaz, F. A. Cheikh and M. Gabbouj, Audio-based Queries for Video Retrieval over Java Enabled Mobile Devices, Proceedings of SPIE (Multimedia on Mobile Devices II), Electronic Imaging Symposium 2006, San Jose, California (USA), 16–20 Jan. 2006.
5. M. Davis and R. Sarvas, Mobile Media Metadata for Mobile Imaging, ICME 2004 Special Session on Mobile Imaging, Taipei, Taiwan, IEEE Computer Society Press, 2004.
6. M. Gabbouj, I. Ahmad, M. Y. Amin and S. Kiranyaz, Content-based Image Retrieval for Connected Mobile Devices, ISCCSP 2006, Marrakech, Morocco, 13–15 March, 2006.
7. B. Gandhi, A. Martinez and F. Bentley, Intelligent Multimedia Content Management on Mobile Devices, Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on, Vol. 3 (2004), pp. 1703–1706, Taipei, Taiwan, 2004.
8. O. Guldogan and M. Gabbouj, Content-based Image Indexing and Retrieval Framework on Symbian based Mobile Platform, European Signal Processing Conference, EUSIPCO 2005, Antalya, Turkey, Sep. 2005.
9. Symbian OS, <http://www.symbian.com/>
10. MS Windows mobile, <http://www.microsoft.com/windowsmobile/>
11. Linux Devices, <http://www.linuxdevices.com/>
12. Lifeblog, <http://www.nokia.com/lifeblog>

13. Nokia, <http://www.nokia.com/>
14. MUVIS, <http://muvis.cs.tut.fi/>
15. J. Keogh, *The Complete Reference J2ME*, Osborne/McGraw-Hill, Feb. 27, 2003.
16. V. Chopra, A. Bakore, J. Eaves, B. Galbraith, S. Li and C. Wiggers, *Professional Apache Tomcat 5*, Wrox, ISBN 0764559028, 17 May, 2004.
17. S. Li, P. Houle, M. Wilcox, R. Phillips, P. Mohseni, S. Zeiger, H. Bergsten, M. Ferris and D. Ayers, *Professional Java Server Programming*, Peer Information Inc., ISBN: 1861002777, August, 1999.
18. S. Kiranyaz and M. Gabbouj, A Dynamic Content-based Indexing Method for Multimedia Databases: Hierarchical Cellular Tree, *Proceedings of IEEE International Conference on Image Processing, ICIP 2005, Genoa, Italy*, pp. 533–536, Sept. 11–14, 2005.
19. M. Partio, B. Cramariuc, M. Gabbouj and A. Visa, Rock Texture Retrieval Using Gray Level Co-occurrence Matrix, *Proceedings of 5th Nordic Signal Processing Symposium*, Oct. 2002.
20. S. Blackburn and D. DeRoure, A Tool for Content Based Navigation of Music, *Proceedings of ACM Multimedia 98, Bristol, England*, Sept. 12–16, 1998.
21. A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith, Query By Humming, *Proceedings of ACM Multimedia 95*, pp. 231–236, 1995.
22. S. Kiranyaz, *Advanced Techniques for Content-based Management of Multimedia Databases*, PhD Thesis, Publication 541, Tampere University of Technology, Tampere, Finland, June, 2005.
23. Deli, <http://cocoon.apache.org/2.1/developing/deli.html>
24. WAP, <http://www.wapforum.org/>
25. S. Kiranyaz and M. Gabbouj, An Interactive Query Implementation over High Precision Progressive Query Scheme, *Proceedings of WIAMIS Workshop 2006, Korea*, 19–21 April, 2006.
26. C. Wong, *HTTP Pocket Reference*, 1st edition published by O'Reilly Media, Inc., ISBN: 1565928628, June 6, 2000.