

# Content-based Interactive Image Retrieval from Java Enabled Mobile Devices

Iftikhar Ahmad, Moncef Gabbouj

**Abstract**—Content-based image retrieval from a mobile device in large image database is challenging. In this paper we present a client-server architecture where a server is running on a personal computer and a client on the device. The client sends content-based query request to the server and the server performs an interactive content-based query and sends the query results to the client. In the query a user of the mobile device define a time interval after that he wants to see the query results. The server generates the query result after the given time. Furthermore the user of the device can get the updated query results at any time during the query operation. This interactive query can avoid un-wanted progressing query results and thus reduce the server query time and memory.

## I. INTRODUCTION

With the generation of digital media by capture and store facility in multimedia devices, there is a need for content management and framework to provide rapid retrieval of digital media items from large archives. Therefore, it has become vital to retrieve desired information expeditiously and efficiently using these devices.

Flickr [13], Facebook [12], Photobucket [16] and YouTube [17] have shown that, manually assigning text to images and videos can be helpful. Although annotated metadata can play an important role for image retrieval, in this paper we focus on Content-Based Image Retrieval (CBIR) [8] from mobile devices [1]. CBIR addresses the problem of accessing the images that bear some certain content and usually relies on the characterization of low level features such as color, shape and texture, all of which can be extracted from the images.

The processing power and memory capacity of mobile phones are increasing all the time but still they are far behind the personal computers (PCs). As CBIR is computational extensive therefore we propose Mobile Multimedia Video Indexing and Retrieval System (M-MUVIS) [14] a client-server architecture where a server runs on the PC and a client on the device. The M-MUVIS server is made of two Java servlets [3] running inside a Tomcat [3] web server which, in fact, performs the content-

based media retrieval on the server side. The first one, M-MUVIS Query Servlet (MQS), performs an efficient image query operation. The second servlet, M-MUVIS Media Retrieval Servlet (MMRS), is used for media format conversion and streaming a media content to the client. The client sends the query to the server on PC, the server performs the query and sends the query results back to the client. To take the advantage of portability and flexibility the client is written in Java 2 Micro Edition for a Java enabled mobile device [4], [5].

With growing image content, an efficient image retrieval technique is deemed required. Specially, for a mobile device user, performing the query can be annoying experience due to the large query processing time [2], [6]. It is therefore, vital to devise a method which not only reduces the query processing time but also performs the query operation without requiring a system equipped with high performance hardware such as fast processors and large memory. In this paper we present an Interactive Query (IQ) [6] for a mobile device which achieves retrieval performance that may not require a superior performing system on the server side and reduce network bandwidth and processing power on the client side. The IQ achieves this by using two parallel processes and a timer on the server side. With the two processes the client can update the query results when ever required. Before IQ, M-MUVIS supported Normal Query (NQ) and Progressive Query (PQ) [2]. In NQ the query results were based on comparing similarity distances of all the images primitives present in the entire database and performing ranking operation afterwards. NQ is costly in terms of processing power and in case of abrupt stopping during the query processes the retrieved query information is lost. The PQ technique on the other hand, is able to provide intermediate results of the on-going query operation by dividing the whole database containing digital images into smaller sub-sets called “clusters”. Then PQ performs the query in the selected cluster. PQ generates the query results after a fix time interval (tp). In large image database with small time interval PQ generates many results that consume lot of memory and the server processing power. The server sends the desired intermediate result (as selected by client) to the client. Sending the intermediate results to the client consume extra network bandwidth, RAM, processing power and battery power of the device. In the proposed IQ the client defines a time interval and the result

Manuscript received October 30 2009.

Iftikhar Ahmad is with Nokia Corporation, P.O.Box 88, FIN-33721, (Tieteenkatu 1, Tampere 33720, Finland); e-mail:iftikhar.ahmad@nokia.com)

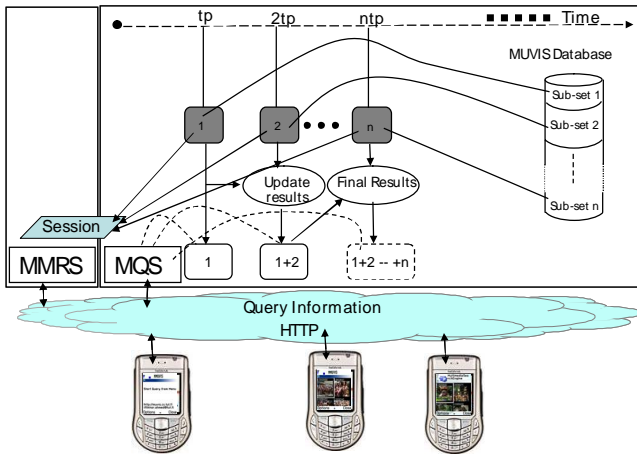
Dr. Moncef Gabbouj is currently a Professor at the Department of Signal Processing at Tampere University of Technology, Tampere, Finland, P.O.Box 553, FIN-33101, Tampere 33720, Finland, (e-mail: moncef.gabbouj@tut.fi).

is generated after that interval and no other intermediate result is generated thus saving the processing power and memory.

Rest of the paper organized as follow. Section II describe progressive query in *M-MUVIS*. Section II is about proposed Interactive query. In section IV experimental results are discussed and summery is in section V.

## II. PROGRESS QUERY IN MOBILE MUVIS

*PQ* operation is achieved over progressive sub-queries (PSQ) [6]. A sub-query is based on the fixed time period, which is fixed (by the client) before starting the query. A sub-query is a partial query operation performed over a sub-set of database items. The database items being used in the sub-query are selected either randomly or by an index structure in the selected database. The architecture of *PQ* in *M-MUVIS* is shown in Fig. 1.



**Fig. 1: PQ architecture in M-MUVIS**

*PQ* has the following features:

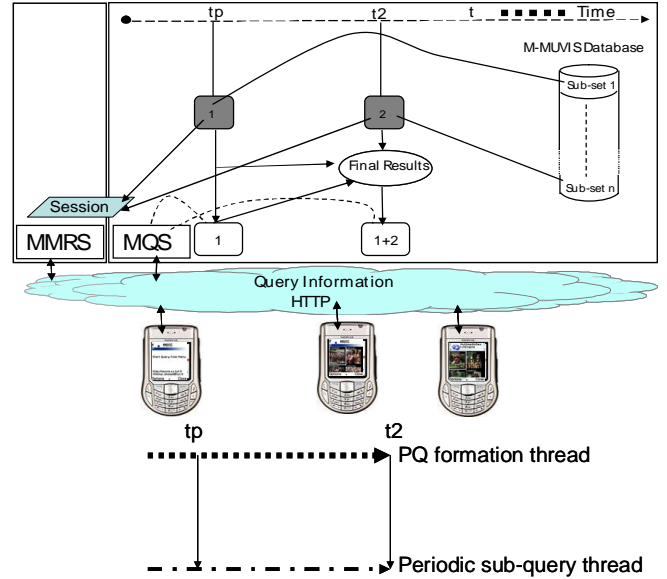
- 1) *PQ* provides intermediate query result after a fix interval of time.
- 2) *PQ* basically generates intermediate query results which are then fused together in a sorted way. The final retrieval result of *PQ* converges to what *NQ* operation produces. This is achieved by fusion operation applied on the intermediate *PSQ* retrievals.
- 3) On an index database the client get the most relevant items in early *PSQ*.
- 4) In large database with small time interval, *PQ* generates many results that consume extra processing power and RAM on server side.

In large image database, extra *PSQ* consume more processing power, network bandwidth and RAM on the mobile device.

## III. INTERACTIVE QUERY IN MOBILE MUVIS

*IQ* has a similar architecture as *PQ* but in *IQ*, *PQ* is divided into two threads (*periodic sub query thread* and *PQ*

*formation thread*) to improve the performance and get better control over the query operations. *Periodic sub-query thread* loads the features from disc to RAM, performs a dissimilarity distance calculation and stores the results in periodic sub-query set. Whereas *PQ formation thread* suspends periodic sub-query thread after a fix interval, apply sub-query fusion and release periodic sub-query thread.



**Fig. 2: Interactive query in M-MUVIS**

In a multi-threaded approach, *IQ* over *PQ* is illustrated in Fig. 2. Idea is to create *PSQ* internally in the query. The intermediate results are not generated for the client rather the client can ask for an update at any time during the query process and get the query results. Once the query operation is completed final result is created and saved on the server side for the client retrieval.

A user can specify the *IQ* settings before initiating a content-based query operation. Since the mobile user does not know before hand about the query completion time due to unknown size of the active database. Hence, the parameters of the *IQ* have been selected accordingly. A fixed time (period =  $T_p$ ) is selected by the client, after that a query result is generated and transmitted to the client. As stated above when the query is completed the final result is created and saved on the server side.

A content-based image query, in a large scale image database from a mobile device, is different from querying it from a PC where the user has more interaction with the system (PC and software) and is watching the ongoing query operation. A mobile user has no idea how the query process is progressing on the server side and how long it will take to complete the query operation. Furthermore, wireless network delay, combined with long query time, is frustrating for the mobile device user. Therefore, the system must be responsive to the user. For this purpose, *IQ* is used

to provide intermediate query results, so that the user can browse them and can retrieve the final query results. In this way, the user does not have to wait a long time for the query operation to be completed. The system will be responsive and interactive. If the user is satisfied with the intermediate results, he can retrieve the original media or start another query. The big advantage of *IQ* is that the user can adjust the query time and he can retrieve the results accordingly.

For efficient retrieval the images in *M-MUVIS* databases are indexed using a recently developed similarity-based indexing scheme called Hierarchical Cellular Tree (HCT) [7]. The main goal of the *HCT* indexing structure is to partition a database in the feature domain into clusters (cell-based), where inter-image distances are minimized, so that in the selected features, similar images are grouped together. In an indexed database, a Query Path (QP) [7] is formed over the clusters (a database subset) of the underlying indexing structure. The *QP* is a special sequence of database items where the most relevant items come first on the path. The advantage of the *IQ* over indexed databases is that the most relevant image items can be retrieved first; this reduces the query time on the server side.

#### IV. EXPERIMENTAL RESULTS

A query-by-example is used to perform the query in a selected database where the feature of the query image is extracted first then the query operation is performed. A set of experiments for image retrieval is carried out to evaluate the performance of the *IQ* in different network technologies such as 3G [9] and Wireless Local Area Network (WLAN). Databases used in the experiments contain 10k and 20K images in JPEG format. Basic visual features such as YUV, HSV, RGB color histograms, and Gray Level Co-Occurrence Matrix (GLCM) [11] as a texture feature, are used on the server side to perform content-based query operations. *M-MUVIS* server is running on P5 1.8 GHz computer with 2048MB RAM memory.

MPEG-7 Average Normalized Modified Retrieval Rank (ANMRR) [10] is used to measure the retrieval performance in *IQ*. It combines both the traditional hit-miss counters; Precision-Recall, and further takes the ranking information into account as given in the following expression:

$$\begin{aligned}
 AVR(q) &= \frac{1}{NG(q)} \sum_{k=1}^{NG(q)} R(k), W=2NG(q) \\
 NMRR(q) &= \frac{2AVR(q) - NG(q) - 1}{2W - NG(q) + 1} \leq 1 \\
 ANMRR &= \frac{\sum_{q=1}^{NQ} NMRR(q)}{NQ} \leq 1
 \end{aligned} \tag{1}$$

where  $NG(q)$  is the minimum number of relevant (via ground-truth) images in a set of  $NQ$  retrieval experiments,  $R(k)$  is the rank of the  $k$ th relevant retrieval within a window of  $W$  retrievals, which are taken into consideration for each query,  $q$ . If there are less than  $NG(q)$  relevant retrievals among  $W$  then a rank of  $W+1$  is assigned for the remaining (missing) ones.  $AVR(q)$  is the average rank obtained from query  $q$ . Since each query item is selected within the database, the first retrieval will always be the item queried and this obviously yields a biased Normalized Modified Retrieval Rank (NMRR) ( $q$ ) calculation and is, therefore, excluded from ranking. Hence the first relevant retrieval ( $R(1)$ ) is ranked by counting the number of irrelevant images a priori. Note that if all  $N(q)$  retrievals are relevant, then  $NMRR(q)=0$ ; and thus the best retrieval performance is achieved. On the other hand, if none of the relevant items can be retrieved among  $W$  then  $NMRR(q)=1$ , corresponding to the worst case. Therefore, the lower  $NMRR(q)$  is the better (more relevant) retrieval for the query  $q$ . Keeping the number of query-by-example experiments sufficiently high, the average NMRR, ANMRR, as expressed in equation (1) can thus be used as the retrieval performance criterion.

We present time statistics along with ANMRR of different *IQ* query operations in *M-MUVIS* over the sample databases. Basically, the mean ( $\mu$ ) and Standard Deviation (SD ( $\sigma$ )), are computed over 20 query operations performed in selected databases. The client query time (CQT) is the waiting period for the query results, recorded by an *M-MUVIS* client. An experimental *CQT* in milliseconds is measured on Nokia N95 [15] and Nokia 5800 [15] as shown in Table 1. We measured the *CQT* by using different networks: the fastest query time is achieved in the WLAN. We have also observed a higher variance in 3G as compared to WLAN networks due to the dynamic nature of wireless networks.

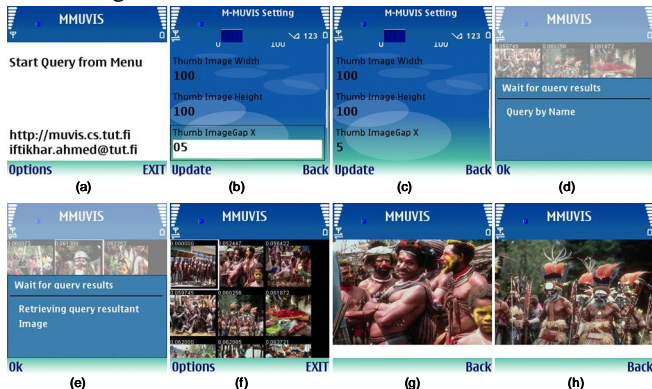
As shown in Table 1 *NQ* takes longer time in large databases, but with the help of *IQ* on *HCT*, *CQT* can be reduced and the most relevant (ANMRR < 0.6) images are retrieved in the predefined time (Tp).

**Table 1: Interactive query results in M-MUVIS**

| Network | <i>IQ</i> | <i>NQ</i> |
|---------|-----------|-----------|
|         |           |           |

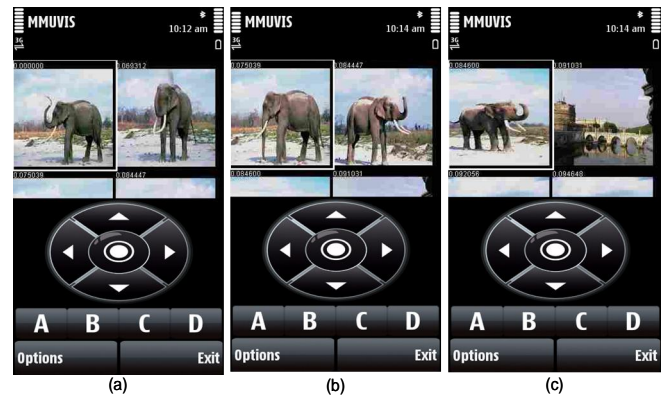
|                       | $CQT$ (ms) |          | ANMRR  | $CQT$ (ms) |
|-----------------------|------------|----------|--------|------------|
|                       | $\mu$      | $\sigma$ |        |            |
| 10,000 image database |            |          |        |            |
| 5800 (3G)             | 1954       | 48       | 0.528  | 8059       |
| N95 (3G)              | 2864       | 1006     | 0.556  | 10919      |
| N95 (WLAN)            | 771        | 28       | 0.5600 | 7891       |
| 20,000 image database |            |          |        |            |
| 5800 (3G)             | 2231       | 242      | 0.5462 | 33740      |
| N95 (3G)              | 2814       | 640      | 0.5444 | 33851      |
| N95 (WLAN)            | 860        | 89       | 0.5442 | 32341      |

Due to the early access of most relevant items  $IQ$  improves the user experience in larger database. As considerable time is used between the client-server communications therefore  $IQ$  has little effect in case of small image databases.



**Fig. 3: M-MUVIS client running on Nokia N90**

Fig. 3 (a) shows the main view of  $M-MUVIS$  client whereas Fig. 3 (b) and (c) shows the setting of  $M-MUVIS$  client. In Fig. 3 (d) the client waits for the query result and in Fig. 3 (e) retrieving a  $QRI$ . The retrieved  $QRI$  is shown in Fig. 3 (f) whereas Fig. 3 (g) and (h) shows original images from the  $QRI$ , after fetching from the server. Fig. 4 shows the query results on Nokia 5800. In Fig. 4 (a) the query image is selected and Fig. 4 (b) and Fig. 4 (c) shows second and third row of the retrieved  $QRI$ .



**Fig. 4: The query results are shown on Nokia 5800.**

## V. CONCLUSION

$PQ$  provides efficient image retrieval from large scale database. However due to its fix duration time interval it generates many results. Whereas  $IQ$  provides an efficient retrieval without generating many intermediate query results in larger image database. Secondly a client user can define the time when he wants to see the query results regardless of database size and the feature selected in the database.

## REFERENCES

- [1] 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, November, 2005.
- [2] I. Ahmad, S. Abdullah, S. Kiranyaz, M. Gabbouj, "Progressive query technique for image retrieval on mobile devices", CBMI, June 21-23, 2005, Riga, Latvia.
- [3] V. Chopra, Amit Bakore, Jon Eaves, Ben Galbraith, Sing Li, Chanoch Wiggers, "Professional Apache Tomcat 5", published by Wrox, May, 2004. ISBN 0764559028.
- [4] H. M. Deitel, P. J. Deitel, Harvey M. Deitel, Paul J. Deitel, "Java How to Program", 5th Edition, published by Prentice Hall, December 1999.
- [5] J. Keogh, "The Complete Reference J2ME", published by McGrawHill OSBORNE Edition. Feb 27, 2003. ISBN: 0072227109.
- [6] S. Kiranyaz, M. Gabbouj, "An Interactive Query Implementation over High Precision Progressive Query Scheme", In Proc. of WIAMIS Workshop 2006, Korea, 19-21 April, 2006.
- [7] S. Kiranyaz, Moncef Gabbouj, "Hierarchical Cellular Tree: An Efficient Indexing Scheme for Content-based Retrieval on Multimedia Databases", IEEE Transactions on Multimedia, vol. 9, no. 1, January 2007, pp. 102-119
- [8] S. Kiranyaz, E. Guldogan, M. Partio, O. Guldogan, M. Gabbouj, "An Extended Framework Structure in MUVIS for Content-based Multimedia Indexing and Retrieval", SETIT 2007, Hammamet - Tunisia, 25-29 Mar. 2007.
- [9] J. Lempiäinen, M. Manninen, Radio Interface System Planning for GSM/GPRS/UMTS, Kluwer Academic Publishers, 2001.
- [10] B. S. Manjunath, P. Salembier, T. Sikora, "Introduction to MPEG-7 Multimedia content description interface", published by John Wiley & Sons, LTD., ISBN 0 471 48678 7
- [11] 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, Trondheim, Norway, October 2002
- [12] "Facebook", <http://www.facebook.com/>
- [13] "Flickr", <http://www.flickr.com/>
- [14] "MUVIS", <http://muvis.cs.tut.fi/>
- [15] "Nokia", <http://www.nokia.com/>
- [16] "Photobucket", <http://photobucket.com/>
- [17] "YouTube", <http://www.YouTube.com/>