Energy Efficient Content-Based Image Retrieval for Mobile Systems

Yu-Ju Hong, Karthik Kumar, and Yung-Hsiang Lu
School of Electrical and Computer Engineering, Purdue University
Email:{yujuhong, kumar25, yunglu}@purdue.edu

Abstract— **This paper presents a method to save energy for mobile systems that perform content-based image retrieval (CBIR). CBIR is a computation-intensive application and a resource-constrained mobile system may save energy by offloading computation to a grid-powered server. We develop three offloading schemes based on different conditions to save energy for CBIR on mobile systems. We implement a CBIR algorithm on an HP iPAQ hw6945 and physically measure the energy savings.**

I. Introduction

Mobile devices such as Personal Digital Assistants (PDAs) have gained popularity with their growing computation capability, flash memory, and various functions. Since most of these devices are equipped with cameras, organization and retrieval of images are important. Content-Based Image Retrieval (CBIR) provides a method to search images based on their contents and is a promising application for these devices. Several studies are dedicated to mobile CBIR [1], [6], [7]. Most of them treat the mobile device as a "thin client", and perform the actual search and feature database maintenance on the server side. Our previous work [7] performs CBIR entirely on the mobile device. This extends CBIR to the cases where network connectivity is limited. However, in other cases, partitioning computation between a mobile device and a grid powered server can save energy on the device. A dynamic decision should be made to decide when and what to offload to the server to save energy.

In this paper, we assume the mobile device is capable of performing CBIR without a server. The important operations of CBIR are identified through experiments; these operations include extracting features from images, loading the feature database to memory, and performing search. We present three offloading options (a), (b), and (c) to reduce energy consumption by executing some operations on a server. Option (a) performs all operations locally on the device; (b) sends the images, performs feature extraction on a server, and updates the feature database on the mobile device to perform search locally; (c) sends the images and the feature database to a server and performs feature extraction and search on the server. In all options, the mobile device maintains the latest database and thus is able to perform CBIR locally next time. The offloading decision is made based on the network bandwidth, the size of the feature databases, the images, and the number of consecutive user queries.

We implement a CBIR program based on ImgSeek [5] on a Linux server and an HP iPAQ hw6945 PDA. Our experimental results show that the bandwidth threshold of computation offloading with a single query is around 60 kB per second. This is achievable in practice. We also show that if the best option is always chosen correctly, we can save 10% to 43% energy for a single query over various bandwidths. For ten queries, the energy savings range from 11% to 45%.

The major contributions of this paper are: (1) It analyzes the energy consumption for mobile CBIR and quantifies the computation and communication energy. (2) We develop three options by considering the effect of loading the feature database, and sending the feature database to a server with multiple queries. (3) We implement our solution and measure the energy savings on a commercial PDA.

II. Related Work

CBIR has been widely studied [3]. An image is represented by a feature vector that contains a set of numerical coefficients. They are used to compare and match images. Jacobs et al. [5] use a feature vector of 60 Haar wavelet coefficients to represent painted images and 40 coefficients to represent scanned images. Their work is implemented in ImgSeek, an open source program that uses these wavelet coefficients.

CBIR is a good candidate for offloading because it is memory intensive and computation intensive. Most existing mobile CBIR *always* offload computation to servers. Ahmad et al. [1] use a Java-based framework that uses a mobile phone (client) to capture the image and perform CBIR on a server. Jia et al. [6] propose an architecture to query the Internet using images taken from mobile systems. Noda et al. in [9] show how CBIR can be used in flower recognition. In [7], we present a system which *always* performs CBIR entirely on the mobile device, for applicaions with limited or no network connectivity (such as a national park).

This paper differs from existing work as the first study on making computation offloading decisions for mobile content-based image retrieval. Existing frameworks on making computation offloading decisions [11], [8], [2], [10], [4] are too general and cannot be applied to CBIR since they cannot use application specific information for the decision making. The decision depends on different conditions such as wireless bandwidth, server speed, etc. Since CBIR is an important application for mobile devices, we leverage application-specific information to develop a framework for offloading CBIR to save energy under *all* conditions.

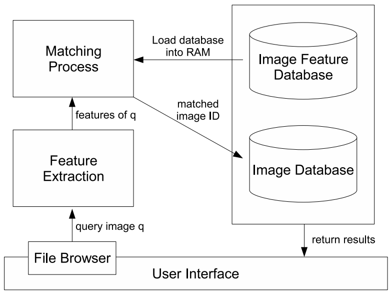

Fig. 1. CBIR program on mobile devices.

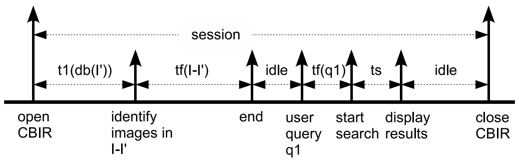

Fig. 2. Operations for a query in a CBIR session.

III. CBIR on Mobile Systems

This section describes how CBIR executes on mobile systems and shows some experimental data of the execution.

A. Execution Time of CBIR

CBIR compares *features*, a set of numerical representations of images to decide the similarity among them. Fig. 1 and Fig. 2 show a general CBIR system on a mobile device and the operations executed for a query. Let I be the image collection and I' be a subset of I; features of images in I' have been extracted and saved in a feature database db. The images in $I - I'$ have not been processed by the CBIR program and are not present in db at this time. We call the images in I' *indexed* images. The database is stored in the flash memory. The operations for a single search are: (1) A user opens the CBIR program and the feature database is loaded into main memory. (2) The program identifies all unindexed images in I, extracts their features, and adds the features to db. (3) The user selects a query image denoted as q. (4) The program extracts the features from q, and compares them with the features in db. (5) The program returns the results (the top 10 matches). We define the time interval from opening to closing the CBIR program as a *session*. A user may have several queries during one session. The total execution time T_{cbir} of a session is

$$T_{cbir} = t_l(db(I')) + t_f(I - I') + n_q \times (t_f(q) + t_s). \qquad (1)$$

Table I lists the symbols and their definitions. The execution time $t_l(db(I'))$ is the time for loading the existing database $db(I')$. The time to extract features from all the unindexed images and add them to the database is $t_f(I - I')$. The user-dependent intervals such as selecting a query are not considered in this paper.

B. Performance of CBIR on Mobile Device

We port an open source CBIR program, ImgSeek [5], to an HP iPAQ hw6945 PDA with a 2GB miniSD card. The iPAQ has an Intel XScale PXA270 416MHz processor and 64MB RAM. We use up to 10,000 JPEG images with an average size of 100kB. We analyze the feature extraction time t_f, database loading time t_l, and search time t_s. *Feature extraction time t_f:* ImgSeek resizes all images to 128x128 resolution, then extracts 60 features. Thus t_f includes resizing and extraction time. Our experiments show that extraction alone is 1.65 seconds for resized images; resizing time is highly correlated to the image resolutions in Fig. 3. For simplicity, in the following sections, we use the average t_f= 2.01 seconds for a 100kB image with 800×640 resolution. The size of the features of an image is 720 bytes; this is relatively small for an average 100kB image in our collection. *Database loading time t_l:* the size of feature database is decided by the number of images. A db of 1000 images is 895kB; a db of 5000 images is 3.62MB. Loading db from the flash memory may take up to 25% of time of a single query with 1000 images. The loading time can be approximated by 0.0065x+1.2 seconds, where x is the number of images. *Search time t_s:* Search time is proportional to the size of the database. Our results show that the search time is merely 1.4% of the feature extraction time when there are 1000 images in the database. Based on the above results, we reformulate (1) by considering only the number of images.

$$T_{cbir} = t_l(|I'|) + t_f \times |I - I'| + n_q \times (t_f + t_s(|I|)). \qquad (2)$$

TABLE I

Symbols and Their Definitions.

symbol	meaning	symbol	meaning
I	image set	t_l	time to load database
I'	indexed image set	t_f	time to extract features
db	feature database	t_s	time to search in db
q	query image	n_q	number of queries
T_{cbir}	time of a session		

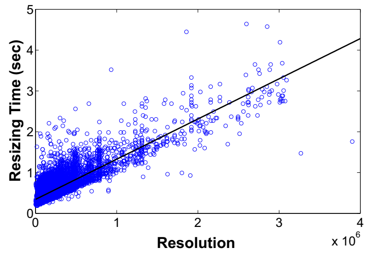

Fig. 3. Resizing time of images with different resolutions.

IV. Computation Offloading

A. Offloading Options

In this section, we introduce three offloading options for different bandwidths, sizes of databases, and number of queries.

Option (a) (local extraction, local search): The energy consumption of the whole session is

$$E_a = P_{co} \times t_l(|I'|) + P_{co} \times t_f \times |I - I'| + n_q \times P_{co} \times (t_s(|I|) + t_f) \quad (3)$$

where P_{co} is the average computing power. Equation (3) is the energy consumption of (2).

Option (b) (remote extraction, local search): The CBIR program loads the existing feature database, sends all unindexed images $I - I'$ to the server, waits and then receives the feature database containing features of images $I - I'$, and loads them into memory. All queries are served locally. Assume the server is α times faster than the mobile device. The total energy consumption is

$$E_b = P_{co} \times t_l(|I'|) + P_u \times \frac{size(I - I')}{BW_u} + P_w \times \frac{t_f \times |I - I'|}{\alpha} + P_d \times \frac{size(db(I - I'))}{BW_d} + P_{co} \times t_l(|I - I'|) + n_q \times P_{co} \times (t_s(|I|) + t_f), \quad (4)$$

where P_u and P_d are the powers of uploading and downloading over a wireless network, BW_u and BW_d are the uploading and downloading network bandwidths. The idle power on the device when waiting for the server is P_w.

Option (c) (remote extraction, remote search): The mobile device is responsible only for sending and receiving data. The mobile device sends $db(I')$ with all unindexed images $I - I'$ to the server. The server extracts features from $I - I'$ and then combines the features with $db(I')$ to obtain $db(I)$. Next, the mobile device sends each query image to the server, and waits for the search results to return. The search result is only a small text file of image IDs and the energy consumption of the search can be ignored. Before the session ends, $db(I - I')$ is sent back from the server such that the most updated database is always in the mobile device; the next session can choose option (a) without needing the server. The total energy consumption of this option is

$$E_c = P_u \times \frac{size(I - I') + size(db(I'))}{BW_u} + P_w \times \frac{t_f \times |I - I'|}{\alpha} + n_q \times (P_u \times \frac{size(q)}{BW_u} + P_w \times \frac{(t_s(|I|) + t_f)}{\alpha}) + P_d \times \frac{size(db(I - I'))}{BW_d}. \quad (5)$$

Option (b) saves local computation of feature extraction by sending the images to the server. Option (c) avoids loading databases by performing all searches on the servers; however, it consumes extra energy by sending $db(I')$. We do not consider local extraction and remote search because it always consumes more energy than the other three options.

B. Analysis of Offloading Decision

Most of the parameters of the equations in the previous section can either be known at runtime, predicted based on the available information, or obtained by measurement. The critical factor in making decision is the uploading network bandwidth BW_u. BW_d is less important because it is usually higher than BW_u. We will use the terms "bandwidth" and "uploading bandwidth" for BW_u interchangeably.

Intuitively, the order of bandwidth requirement for the options is $(a) < (b) < (c)$, since more parts of the execution have been shifted to the server in this order. To clearly examine the required bandwidth for each option, we rewrite E_b and E_c and define E_{th1}, E_{th2}, and E_{th3} as follows.

$$E_b = E_a - E_{th1}, \quad (6)$$

$$E_c = E_b - (E_{th2} + n_q \times E_{th3}), \quad (7)$$

E_{th2} is defined as $E_b - E_c$ when n_q is zero; using E_{th2}, E_{th3} is further defined for the difference between E_b and E_c when n_q is not zero.

TABLE II
POWER PARAMETERS OF THE MOBILE DEVICE.

Parameter	Operation	Avg. Power
P_w	Idle	298.6mW
P_{co}	Computation	866.0mW
P_u	Upload	1279.5mW
P_d	Download/On	1571.4mW

In (6), if E_{th1} is positive, (b) is a better choice than (a) because the overhead of transferring images is compensated by reducing the computation time on the mobile device. We named the bandwidth when E_{th1} is zero as B_{th1}, the bandwidth threshold for offloading. It can be computed from a given feature extraction time and $|I'|$. From (7), (c) consumes less energy than (b) when (1) E_{th2} and E_{th3} are both positive; (2) one of E_{th2} and E_{th3} is negative but $E_{th2} + n_q \times E_{th3} \geq 0$. When $E_{th2} > 0$, the energy consumed by transferring the existing database $db(I')$ can be compensated by eliminating the need to load $db(I)$ on the mobile device. This happens when $|I'|/|I|$, the *indexed ratio* is low. If E_{th3} is larger than zero, it indicates that the advantage of uploading the query image and search outweighs the local feature extraction and search. Under this condition, the energy saved by (c) over (b) increases with the number of queries.

We set B_{th2} and B_{th3} to be the bandwidth thresholds when E_{th2} and E_{th3} equal to zero, respectively. B_{th2} is determined by the indexed ratio. B_{th3} is partly determined by the size of the query images which are unknown at decision time. We use the average size of images in I as the expected size of the query images. In this case, B_{th3} differs from B_{th1} only by t_s and the energy of downloading $db(I - I')$. Since t_s is only 1.4% of t_f and energy of downloading the database is much lower than the uploading counterpart, B_{th1} is almost equal to B_{th3}. Thus, when the bandwidth is lower than B_{th1}, (a) should be chosen. If the bandwidth is higher than both B_{th1} and B_{th2}, it indicates that the bandwidth is sufficient to allow most of the operations to be done on the server, so (c) is the best choice. If the bandwidth is higher than B_{th1}, but lower than B_{th2}, whether (b) or (c) should be chosen depends on n_q. If n_q is larger enough to let $n_q \times E_{th3}$ compensate E_{th2}, (c) should be chosen.

V. EXPERIMENTAL RESULTS

A. Experiment Setup

We measure the actual energy consumption on the iPAQ using the 3 options. We connect a 0.25Ω resistor in series with the battery of the iPAQ and measure the voltage drop across the resistor with a data acquisition card installed on a separate computer. The sampling frequency is 10kHz. We turn the wireless network on for all experiments. Table II lists the power parameters for the iPAQ for reference. The ratio of downloading over uploading bandwidths is 2.8 in our experiment. We uses a Linux server with an Intel Xeon 2.33GHz processor and 8GB memory. The server-client performance ratio α is 300.

B. Results and Discussions

1) Local or Remote Feature Extraction: Fig. 4 shows how the available network bandwidth affects E_a and E_b for a single

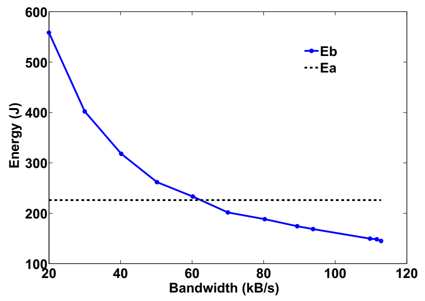

Fig. 4. E_a and E_b at different bandwidths when $|I'| = 900$, $|I| = 1000$, and $n_q = 1$.

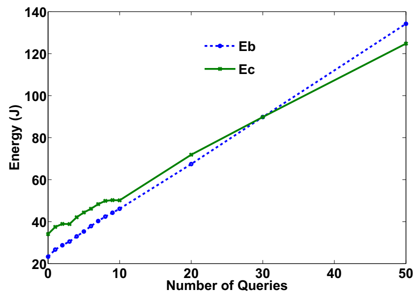

Fig. 5. E_b and E_c when the indexed ratio is 0.99 and the bandwidth is 80kB/s.

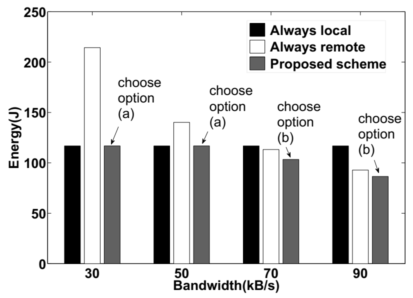

Fig. 6. Energy consumptions compared for a single query.

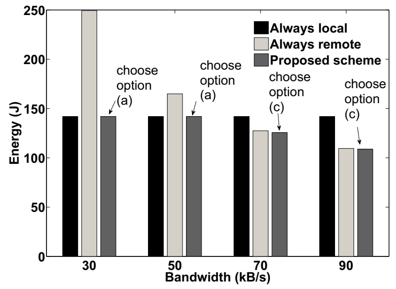

Fig. 7. Energy consumptions for 10 queries.

query. The bandwidth threshold B_{th1} is around 60kB per second. Hence, if the available bandwidth is higher, (b) should be chosen over (a). When the bandwidth is only 20kB per second, (b) consumes 2.47 times more energy than (a).

2) Indexed Ratio: We observe that (b) is more energy efficient than (c) when the bandwidth is 80kB per second for a single query under different indexed ratios. In this case, B_{th2} is 180kB per second, and it is beyond the maximum bandwidth. When the indexed ratio is 0.99, (b) consumes only 50.8% energy of that of (c). However, as the indexed ratio decreases, more images need to be sent to the server, leading to an increase in the total amount of energy. Thus the energy of sending $db(I')$ compared to the total energy consumption is insignificant. If the CBIR application is used frequently (db updated frequently), keeping the indexed ratio low, (b) outperforms (c).

3) Multiple Queries: Fig. 5 shows a case where E_c is less than E_b even the bandwidth is lower than B_{th2} and the indexed ratio is very high, under multiple queries. We set the indexed ratio to be 0.99 for $|I| = 1000$. The bandwidth is 80kB per second. E_c is greater than E_b at $n_q = 1$, but when n_q exceeds 30, E_c becomes smaller than E_b. Hence, if the CBIR usage (n_q) can be known or estimated, it is possible to reduce the expected energy for CBIR sessions.

In Fig. 6, we show the energy consumptions of always local (a), always offloading (c), and the adaptive method choosing (a), (b), or (c), when $|I|$ is 500, and $|I'|$ is 50, and n_q is one. The results show that choosing the best options can save 10% to 43% energy over various bandwidths when compared to the worst of always local and always offloading. Fig. 7 shows the energy consumptions when n_q is 10. We can save between 11% and 45% energy in this case.

VI. Conclusions

We evaluate the performance of CBIR on a mobile device and provide a method to adaptively offload computation in order to save energy. We validate our method using physical measurement of energy consumption. We show the conditions under which offloading is beneficial. The energy savings range from 10% to 43% for a single query, and from 11% to 45% for ten queries.

Acknowledgments

This work is supported in part by NSF CNS-0347466 and CCF-0541267. Any opinions, findings, and conclusions or recommendations in the project are those of the authors and do not necessarily reflect the views of the sponsor.

References

[1] I. Ahmad and M. Gabbouj. Compression and Network Effect on Content-Based Image Retrieval on Java Enabled Mobile Devices. In *Finnish Signal Processing Symposium*, 2005.

[2] R. K. Balan, M. Satyanarayanan, S. Park, and T. Okoshi. Tactics-Based Remote Execution for Mobile Computing. In *International Conference on Mobile Systems, Applications, and Services*, pages 273–286, May 2003.

[3] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image Retrieval: Ideas, Influences, and Trends of the New Age. *ACM Computing Surveys*, 40(2):60, 2008.

[4] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic. Adaptive Offloading for Pervasive Computing. *IEEE Pervasive Computing*, 3(3):66–73, July- September 2004.

[5] C. E. Jacobs, A. Finkelstein, and D. H. Salesin. Fast Multiresolution Image Querying. In *International Conference on Computer Graphics and Interactive Techniques*, pages 277–286, 1995.

[6] M. Jia, X. Fan, X. Xie, M. Li, and W.-Y. Ma. Photo-to-Search: Using Camera Phones to Inquire of the Surrounding World. In *International Conference on Mobile Data Management*, pages 46–46, 2006.

[7] K. Kumar, Y. Nimmagadda, Y.-J. Hong, and Y.-H. Lu. Energy Conservation for Content-based Image Retrieval on Mobile Devices. In *International Symposium on Low Power Electronic Design*, 2008.

[8] Z. Li, C. Wang, and R. Xu. Computation Offloading to Save Energy on Handheld Devices: a Partition Scheme. In *International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, pages 238–246, 2001.

[9] M. Noda and H. Sonobe. Cosmos: Convenient Image Retrieval System of Flowers for Mobile Computing Situations. In *IASTED Conference on Information Systems and Databases*, pages 25–30, 2002.

[10] R. Wolski, S. Gurun, C. Krintz, and D. Nurmi. Using Bandwidth Data to Make Computation Offloading Decisions. In *High-Performance Grid Computing Workshop*, April 2008.

[11] C. Xian, Y.-H. Lu, and Z. Li. Adaptive Computation Offloading for Energy Conservation on Battery-Powered Systems. In *International Conference on Parallel and Distributed Systems*, December 2007.