Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Karthik Kumar

Entitled Energy Conservation for Content-based Image Retrieval on Mobile Devices

For the degree of Master of Science in Electrical and Computer Engineering

Is approved by the final examining committee:

Y. H. Lu
Chair

M. Thottethodi

V. Raghunathan

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Y. H. Lu

Approved by: M. J. T. Smith July 22, 2008
Head of the Graduate Program Date

Graduate School Form 20
(Revised 10/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL

Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:

Energy Conservation for Content-based Image Retrieval on Mobile Devices

For the degree of Master of Science in Electrical and Computer Engineering

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Executive Memorandum No. C-22,* September 6, 1991, *Policy on Integrity in Research.**

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Karthik Kumar
Signature of Candidate

June 18, 2008
Date

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/c_22.html

ENERGY CONSERVATION FOR CONTENT-BASED IMAGE RETRIEVAL ON MOBILE DEVICES

A Thesis

Submitted to the Faculty

of

Purdue University

by

Karthik Kumar

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

August 2008

Purdue University

West Lafayette, Indiana

This work is dedicated to my parents and brother.

ACKNOWLEDGMENTS

I would like to thank my adviser Professor Yung-Hsiang Lu for having taught me how to do research, and for his constant guidance, mentorship and support. Thanks are also due to members of my committee Professor Vijay Raghunathan and Professor Mithuna Thottethodi for reviewing my work.

I would like to extend my thanks to my friends and colleagues in the HELPS research group: Yamini Nimmagadda, Yu-Ju Hong, Changjiu Xian, Jackie Feng, Christian Berrios, Eddie Pettis, Jeff Brateman and Douglas Herbert. I appreciate the help of Yu-Ju Hong and Dave Azpell with the energy measurements in this work.

I would like to thank my parents and brother for the many sacrifices they have made for me through the years. Thanks are also due to Shruthi Sudheer for her understanding and to all my friends for their encouragement.

I want to thank the financial support from NSF CCF-0541267. Any opinions, findings, and conclusions or recommendations in this thesis do not necessarily reflect the views of the sponsor.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Kumar, Karthik M.S.E.C.E, Purdue University, August, 2008. Energy Conservation for Content-based Image Retrieval on Mobile Devices. Major Professor: Yung-Hsiang Lu.

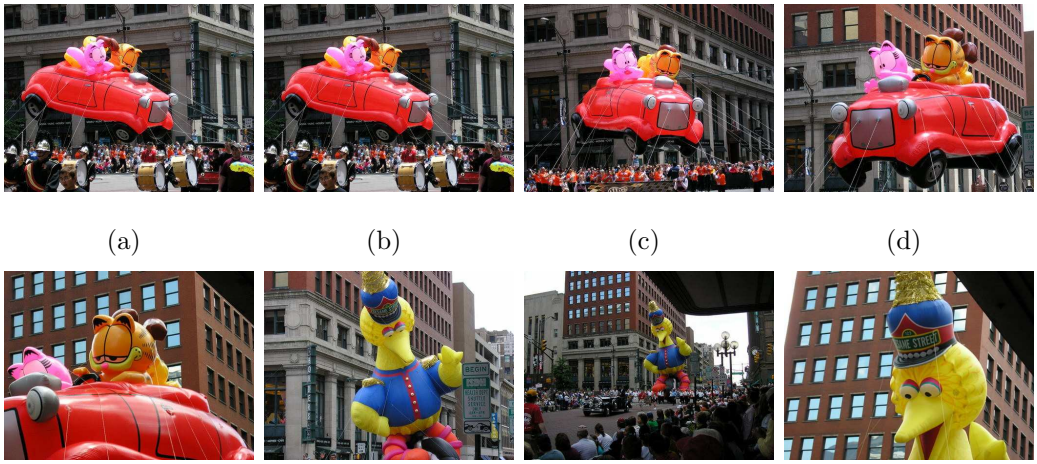
Battery-powered mobile systems such as PDAs (personal digital assistants) and mobile phones play an increasing role in handling visual contents such as images. Thousands of images can be stored in a mobile system with the advances in memory technology; this creates the need for better organization and retrieval of the images. Content Based Image Retrieval (CBIR) provides a method to retrieve images based on their contents. In CBIR, images are represented and compared by high-dimensional vectors called features. Loading these features into memory and comparing them consumes a significant amount of energy. In this thesis, we present the first study on energy conservation for CBIR on a mobile system. We develop an adaptive loading scheme to save energy for CBIR. Our method reduces the features to be loaded into memory for each query image. The reduction is achieved by estimating the difficulty of the query. If the images are dissimilar, fewer features are sufficient; less computation is performed and energy can be saved. For each query image, a similarity index is calculated to determine the features' length for meeting a target accuracy. Further, we consider the effect of consecutive user queries and show how features can be "cached" in memory to save energy. We implemented this algorithm on an HP iPAQ hw6945 and measured the energy savings. For a collection of 5000 images, we obtained average energy reduction of 61.3% compared to an existing CBIR implementation.

1. INTRODUCTION

Most battery-powered mobile systems, such as cellular phones and PDAs, have built-in cameras and they play an increasing role in handling visual contents such as images. With the improvement in storage technology, it is possible to store several gigabytes of images on these systems. This results in a need for better organization and retrieval in these image collections. Retrieval by image names becomes difficult as the number of images increases, and often the names may not describe the contents of the images. These systems usually have no keyboards or miniature keyboards and keyword-based search is inconvenient. Content-Based Image Retrieval (CBIR) provides a different, possibly more convenient approach to find images. However, CBIR is computationally intensive and energy is limited on these systems. CBIR is performed by analyzing images and extracting their features. These features are represented by high-dimensional numerical vectors, and comparisons between images are performed by matching their feature vectors. The selection and comparison of features determines the accuracy, speed, and energy consumption of CBIR. Figure 1.1 shows an example of image retrieval based on content.

Existing CBIR assumes that the mobile systems are "thin clients"— they transmit query images to servers. Feature extraction and matching are performed at the servers. However, the mobile systems' large collections of images may be unavailable at the servers. Transmitting several gigabytes of images through wireless networks for searching one image is unrealistic. Moreover, the mobile systems may not have access to wireless networks (for example, inside a national park). Hence, it is desired to be able to perform CBIR on mobile systems themselves.

Existing CBIR implementations load the entire image collections (also called an "image database") into memory (RAM) before comparing the features. The mobile



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 1.1. An example of CBIR. (a) is the query image of a float in a parade in a collection of images and (b)-(h) are the top matches returned based on contents.

systems have limited resources: energy, processor speed, and memory. Even though the capacities of flash memory have been steadily rising in the past few years, most mobile systems have no more than 64MB memory. This is because energy is limited, and the amount of power needed to hold data in RAM increases with the size of RAM. Flash memory requires almost the same amount of power regardless of sizes. Large image collections can be stored in the flash memory, but current technology does not permit these images to be accommodated into RAM. Even if a mobile system has sufficient RAM, loading the entire image collection into RAM takes time and energy. Thus, it is necessary to develop the technology for CBIR on mobile systems with large flash memory and much smaller RAM. Unfortunately, no previous study has been devoted to analyzing and reducing energy consumption for CBIR on a mobile system. We believe this is the first study on energy conservation for running CBIR completely on a mobile system. This work uses "memory" and "RAM" interchangeably; we use "flash" for non-volatile solid-state storage.

In this thesis, we present a method to save energy for mobile CBIR by adaptively selecting the features to be loaded into memory. The features are selected based on (1) the estimated difficulty of the query, and (2) the features of previous queries.

A search is "difficult" if the query image is similar to many images in the collection; more features are needed to distinguish the images. This consumes time and energy. If a search is easier, fewer features are loaded into memory for comparison; this saves time and energy. For example, it would be more difficult to retrieve the closest and expected matches for a given image of a forest from a database of largely similar forest images than from a database of cars with very few similar images of forests. This increased difficulty requires more features to be matched in order to achieve better accuracy. Furthermore, we cache features in memory for subsequent similar queries to save energy.

Before a query, the maximum length of the features required are pre-computed for the collected images. The features are summarized to determine the similarity among a query image and the collected images. When a query image is received, its features are computed and compared with the feature summary to determine the length of features for comparison. A user can specify the desired accuracy; higher accuracy leads to longer features and more energy consumption. If the search result is unsatisfactory, the user can raise the accuracy and additional features are compared.

Figure 1.2 summarizes the work in this thesis. The rest of this thesis is organized as follows: Chapter 2 describes the related work for mobile CBIR. In Chapter 3, we establish a relationship between energy consumption and feature length, and provide a method to adaptively reduce the feature lengths in order to save energy. Chapter 4 describes three techniques we provide to save energy for mobile CBIR-selective loading, adaptive loading, and feature caching. In Chapter 5, we describe our implementation and experimental setup. We implement our algorithms on an HP iPAQ 6945 PDA with a 416 MHz Intel XScale processor and 64 MB of memory, using a 2GB mini SD flash card for storage. We perform CBIR with different sizes of image collections, different query images and different numbers of query images.

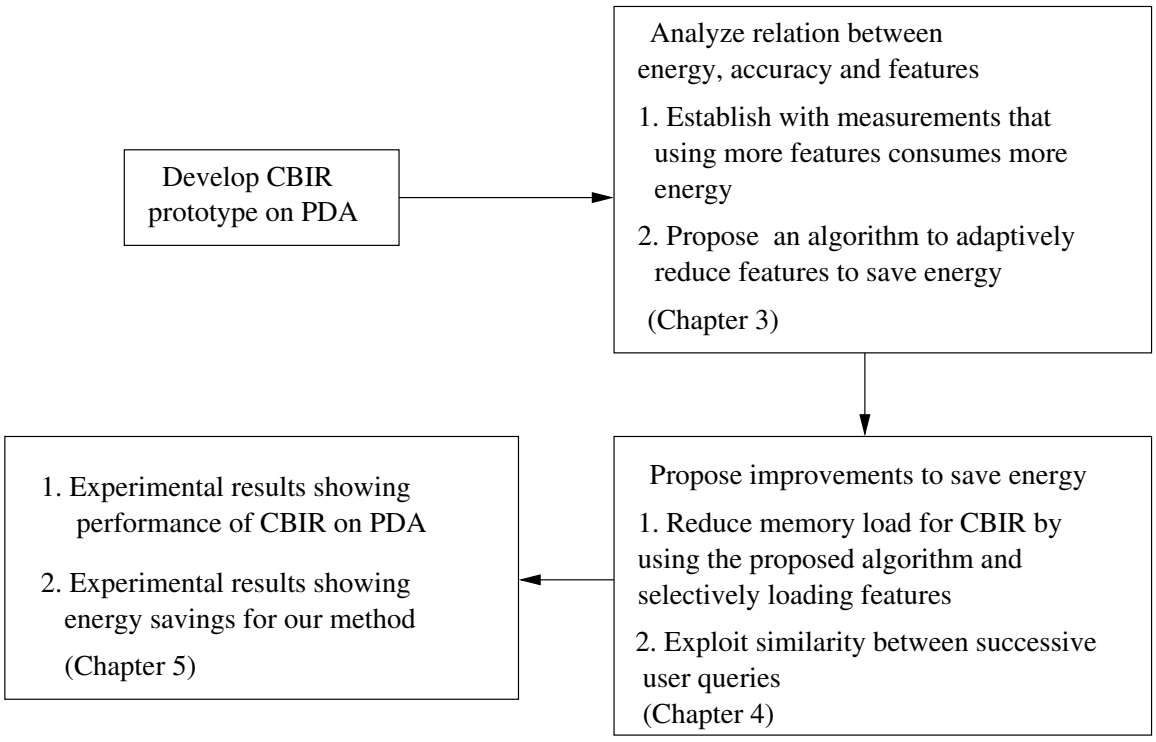


Fig. 1.2. Flow of research in this thesis- implementation and energy conservation for CBIR.

We characterize the performance of our system and measure the energy savings. Our system saves 61.3% energy on average for a single image query, while maintaining an accuracy of 80% for an image collection of 5000 images.

2. RELATED WORK

CBIR is usually performed by comparing images' "features": numerical representations of the images. The features can be determined in many ways, such as color, texture, and shape. CBIR has been widely studied [1]. Several CBIR algorithms use image properties. A typical CBIR algorithm extracts features from all the images offline and stores these features in index files on disks or in flash. Most CBIR programs load the entire index files into memory before comparing the query image's features. CBIR requires large amounts of memory operations and computation in order to return accurate results.

2.1 Memory and Accuracy in CBIR

It is important to analyze the memory performance of CBIR because of the latency involved in accessing secondary storage such as flash memory. Forster in [2] suggests using "filtered retrieval" to reduce time and RAM requirements for CBIR. A filtering mechanism is used to reduce the search space and accommodate all the features in RAM; this makes retrieval faster. Robles et al. in [3] present a parallel implementation of a CBIR system which has been designed for a shared memory system. Davidson at al. in [4] present a parallel technique for searching a memory-resident signature tree (S-tree).

Since CBIR involves finding similar images, it is important to analyze and quantify accuracy in CBIR. The length of the feature vectors used for comparison determines the accuracy in CBIR. Rudinac et al. in [5] analyze the effects of the lengths of feature vectors and observe that shorter vectors do not necessarily degrade accuracy, but reduce processing time. Quantifying accuracy in CBIR also remains a challenge

since the accuracy of the search results are heavily dependent on the query image and the image collection. Gunther et al. in [6] suggest a methodology for developing test images in CBIR. They systematically modify test images which differ in terms of shape, color, texture, etc. Since the test images differ in many properties, they are not biased towards any particular CBIR algorithm. French et al. in [7] give a scoring scheme for query results in CBIR. They define a set of images which are similar to a query image, and calculate the score for the query based on the appearance of these similar images in the search results.

2.2 ImgSeek

We choose ImgSeek, an open source CBIR program as the basis for our work. ImgSeek uses a feature vector of 60 Haar wavelet coefficients for painted images and 40 coefficients for scanned images [8]. The first step in ImgSeek is to pre-process the image collection to form a feature database. Each image is resized to a 128x128 bitmap in the RGB color space. ImgSeek converts the pixels from the RGB color space to the YIQ color space. The Y represents the luminance, and the I and Q components represent the chrominance information. This separation of luminance (brightness) from chrominance (color) improves the quality for for image comparison in CBIR [8]. Each color is then decomposed using the Haar wavelet decomposition. The decomposition is similar to a successive averaging process, which preserves detail coefficients to represent the original image. At the end of the decomposition, the overall average color is present in the (0,0) position along with 16384 wavelet coefficients. These coefficients are then sorted by magnitudes and only the locations of the largest 60 coefficients per color are saved are the feature vector for that image. These feature vectors are organized using inverted indexing, and are compared and matched for image retrieval.

2.3 Existing Mobile CBIR

CBIR on mobile devices is a promising application because most mobile devices are equipped with cameras and have abundant storage. All existing studies examine the use of client-server architecture to perform content-based image retrieval on a mobile device. The image is captured on the mobile device (client) and transmitted to a server. The server then performs CBIR and returns the results to the client. Ahmad et al. [9] present a Java-based framework that uses a client to capture the image and a server to perform matching. Yeh et al. [10] propose a system that first queries an image database (on a server) by using a photograph from a mobile device. After a match is found, the system then uses the keyword obtained with the initial query to perform a Google image search. Jia et al. [11] propose an architecture to query the web directly using images taken from mobile devices.

Studies also show specific applications where CBIR on mobile devices becomes particularly important. Sonobe et al. in [12] analyze the importance of mobile CBIR for fish image retrieval. Noda et al. in [13] show how CBIR can be used to identify unknown flowers. Such applications can often be used in places with limited or no network connectivity (such as a national park). However, both papers suggest an infrastructure that relies on the wireless network since they assume a client-server framework.

2.4 Thesis Contributions

Our contributions [14] differ from the above systems in the following aspects:

(1) We present a system in which image retrieval is performed *entirely* on the mobile device. The image can be captured using the built-in camera, and feature extraction and search can be performed on the device itself.

(2) Energy is limited in mobile devices. Our system is the first to characterize the relation between accuracy and energy for CBIR on the mobile system. We adaptively reduce the computation required for CBIR to save energy.

(3) We analyze and quantify the features loaded into memory for CBIR. In mobile devices, the features are stored in flash and the energy required to read from flash into memory is significant. We adaptively reduce the features that are loaded in memory to save energy.

(4) Our system is the first to analyze the energy consumption over multiple queries. We show an energy-efficient CBIR implementation by reusing features in memory.

3. ADAPTIVE FEATURE REDUCTION

In this chapter, we characterize the energy consumption of CBIR. We then analyze the relationship between accuracy, energy and features in CBIR. We establish the relationship between feature lengths and energy in CBIR by physical measurements. We then introduce a term called the similarity index used to adaptively reduce the feature lengths in CBIR in order to save energy, while maintaining a target accuracy.

3.1 Energy Characterization of Mobile CBIR

We use ImgSeek as an example for our analysis, but our work is general and can be applied to other CBIR algorithms. For CBIR, a feature vector containing m elements [f_1, f_2, ..., f_m] is used to represent an image in each color (40 is suggested for m in [8]). Each of these elements α ($1 \leq \alpha \leq m$) can assume a value in the following range [l_α, h_α]. ImgSeek uses [l_α, h_α] = [1, 16384] for all α. Let i be an image in the collection I, $i \in I$, with feature vector $f(i)$. Similarity between images is given by the number of elements in the feature vector that are common between them. Inverted indexes are used to indicate the set of images with a particular value in the features. Let $R(v)$ be the set of images whose features have an element of value v:

$$R(v) = \{i \in I : \exists s, 1 \leq s \leq m, f_s = v, f_s \in f(i)\}. \tag{3.1}$$

The inverted indexes are calculated *offline* for the entire image collection and stored in flash memory. When the CBIR application starts, all the features are loaded into memory and consume energy $E_d(|I|)$, where $|I|$ is the size of the image collection. When an image is queried, its m features are extracted online and consume energy E_q. The inverted indexes are used to assign scores to all the images that share each of

the m features with the query image; this consumes energy $E_s(|I|)$. The value of m used is fixed and shall be denoted by m_o (=40). The energy consumed by CBIR can be modeled as $E_d(|I|) + E_q + E_s(|I|)$. $E_d(|I|)$ and $E_s(|I|)$ depend on the size of the image collection. This is shown in Figure 3.1 (a). For n queries the energy consumed by CBIR is

$$E_{cbir} = E_d(|I|) + n \times (E_q + E_s(|I|)). \tag{3.2}$$

The energy to load the indexes is consumed only once. ImgSeek loads *all* features to memory *before* search and reduces the time a user has to wait for the results after a query. This is showed as t_{cbir} in Figure 3.1(a). Two problems arise when the entire indexes are loaded into memory. First, the available memory in a mobile system is limited. Second, even if a mobile system has a sufficient amount of memory, loading the indexes takes time and consumes energy.

We propose to load only the features that are relevant to the query image *after* the query image is selected and its features are extracted. In our method, loading indexes and performing comparison are interleaved, as shown in Figure 3.1(b). The energy consumption is expressed as E_{int} and it includes three terms: (1) E_q to extract the query image's features, (2) $E_{d-a}(m, |I|)$ to load the features needed from flash to memory, and (3) $E_s(m, |I|)$ to perform comparison. We use t_{int} to represent the time to perform CBIR for our interleaved scheme. The additional wait time for the user in our scheme is the difference t_{cbir} - t_{int}; we show in our experiments (Chapter 5) that this difference is shorter than 2 seconds.

For comparing n images, E_{int} is

$$E_{int} = n \times (E_q + E_{d-a}(m, |I|) + E_s(m, |I|)). \tag{3.3}$$

In the following section, we examine the significance of the feature length m in equation (3.3) in determining the accuracy and energy consumption of mobile CBIR. The total energy consumption is largely dependent on the sum of the terms

Table 3.1

Symbols and their definitions.

symbol	meaning
I	image collection
q	query image
m	feature length
m_o	maximum feature length
ρ	similarity index
p	quality of search results
E_{cbir}	total energy of original scheme
E_{int}	total energy of our scheme
E_q	energy to extract features
E_s	energy to search
E_d	energy to load features in original scheme
E_{d-a}	energy to load features in our scheme
t_{cbir}	time to perform CBIR
t_d	time to load features
t_q	time to extract features
t_s	time to search in current database
n	number of queries

$E_{d-a}(m, |I|)$ and $E_s(m, |I|)$. The feature length plays an important role in determining the accuracy of CBIR as it is the metric used to distinguish images. We analyze the relation between these various parameters and show how we use our analysis to save energy.

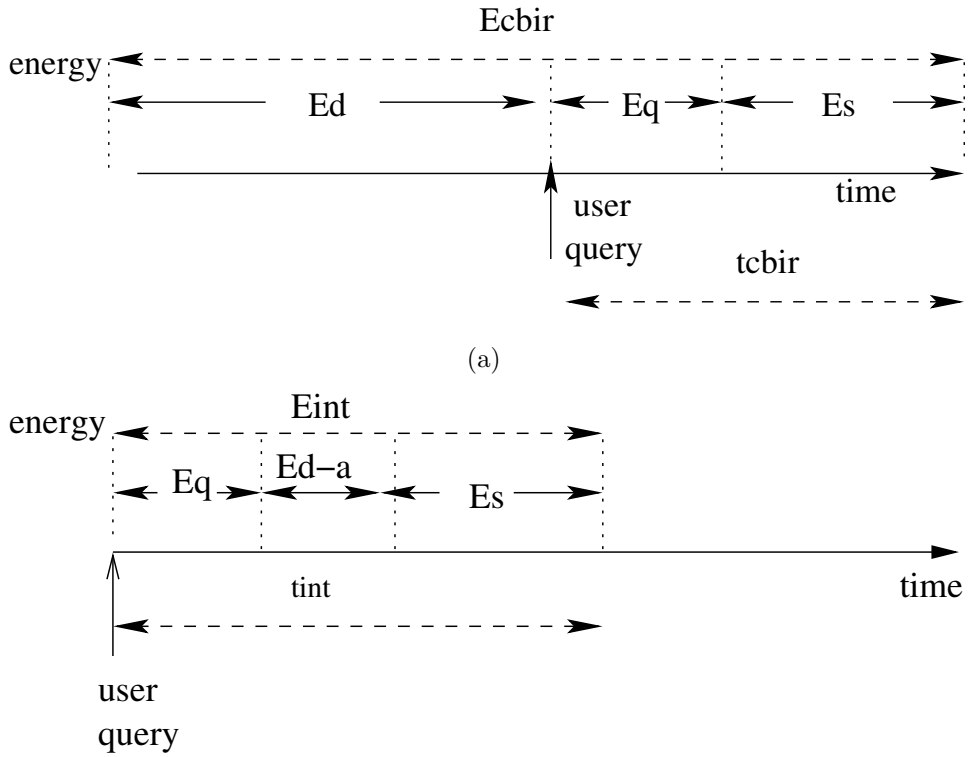


Fig. 3.1. Energy and time in CBIR: (a) existing implementation is compared to (b) proposed interleaving scheme.

3.2 Features, Accuracy and Energy in CBIR

The accuracy of CBIR is affected by several factors, including

- Selection of appropriate features: In this thesis, we use ImgSeek as an example to determine the features but our approach is general and not restricted to a particular algorithm.
- Length of the features used: For a given set of features, if more of them are used, images can be distinguished more accurately. However, this takes longer and consumes more energy.

Fig. 3.2. Content Based Image Retrieval (CBIR) implemented on HP iPAQ hw6945.

- Similarities among the images: If the query image is dissimilar from the existing collection of images, most images can be excluded from the comparison quickly without reducing the accuracy.

3.2.1 Feature Length and Energy

We port ImgSeek to perform CBIR on an HP iPAQ hw6945. We resize the images in the collection to a 128 x 128 bitmap, and extract features using the Haar wavelet decomposition described in Chapter 2. We then measure the energy consumed by an HP iPAQ 6945 while performing CBIR. We obtain the energy consumption by measuring the current samples across a 0.25 *ohm* resistor between the iPAQ and its battery. The experimental setup is described in detail in Chapter 5. Figure 3.3 shows that the energy consumption increases with longer features for more images. For example, in a collection of 2000 images, if 60 coefficients are used, nearly 40%

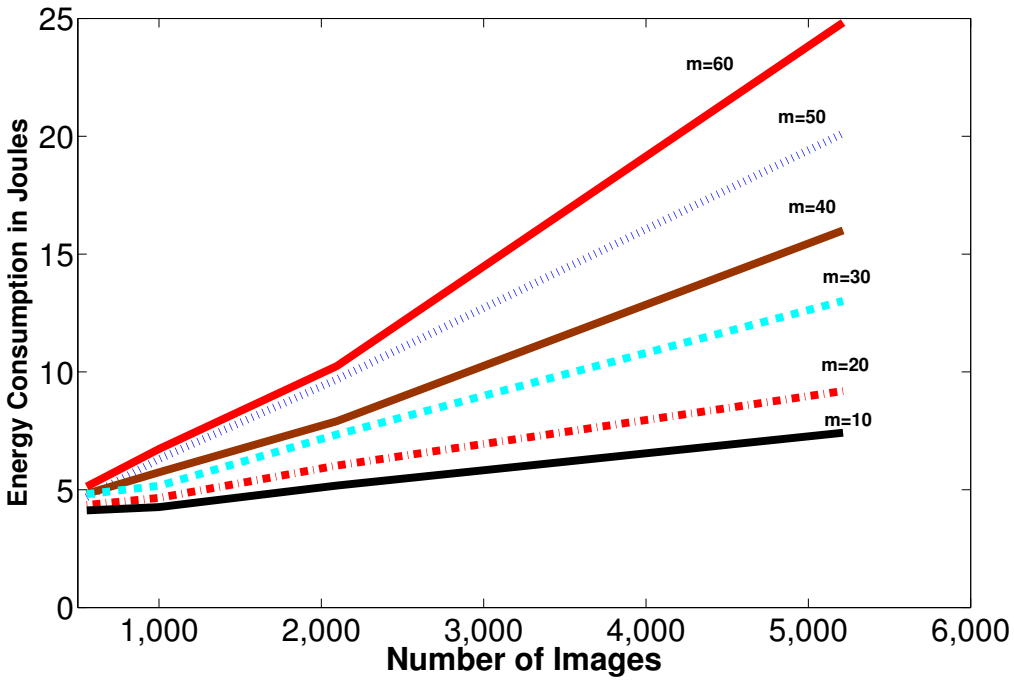


Fig. 3.3. Energy consumption for different feature lengths and sizes of the image database.

more energy is required compared to 30 coefficients. All 16384 features are loaded into memory during this measurement. Figure 3.3 indicates that reducing the feature length reduces the energy consumed by CBIR.

3.2.2 The Similarity Index

If many images in the database are similar to the query image, longer features are used to distinguish them. We quantify similarities among the database and the query image using the following procedure. If two images have the same values in many features, the two images are similar. The feature vector for an image $i \in I$ is $f(i)$, $|f(i)| = m_o$. Let q be a query image with features $f(q)$; the query image does not have to be inside the database. An image j, compared to i, is considered more

similar to q if more features of j and q share the same values than those of i and q do. We can define inverse indexes to indicate the number of images with a particular value in the features. Let $R(v)$ be the set of images whose features have an element of value v, as defined in equation (3.1). Let f denote the complete m_o features of the query image i, $f_j \in f(i)$. The similarity index is defined as

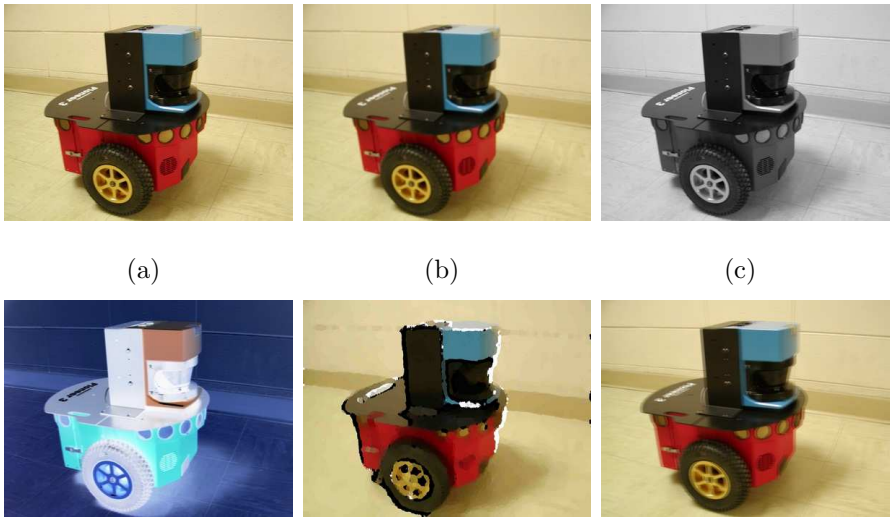
$$\rho = \frac{\sum_{1 \le j \le m_o} |R(f_j)|}{m_o |I|}, \tag{3.4}$$

The numerator is the sum of the images whose features have at least one element equal to f_s. This sum is normalized by the number of elements in the features and the number of images in the database. A larger ρ suggests a higher probability that the database contains many images similar to q and more features are needed to distinguish them. If ρ is small, we only need to compare the first m elements, $(m < m_o)$. To reduce the run-time overhead for computing ρ, $\frac{R(v)}{m_o|I|}$ can be calculated in advance for each v. Even if the image collection spans all possible features, storing $\frac{R(v)}{m_o|I|}$ will require only $16384 \times 2 = 32$ kB (16384 features, 2 bytes per feature, for up to $2^{16} - 1 = 16535$ images).

To define accuracy, we use p to indicate the quality of the search results. The value of p is given as the sum of $p1$ and $p2$. For a query image q, A is a set of similar images. Only the top k $(k > |A|)$ images returned by CBIR are considered for calculating the accuracy. All images ranked $k + 1$ or worse are ignored. The top k images form a sequence $[u_1, u_2, ..., u_k]$. If an image in A is ranked γ $(1 \le \gamma \le k)$, γ is added to $p1$. The $\delta(b)$ function is one if b is true and zero if b is false.

$$p1 = \sum_{i \in A} \gamma \times \delta(i = u_\gamma). \tag{3.5}$$

If an image in A is not among the top k images, a penalty τ is added to $p2$. Apparently, τ has to be larger than k.



(a) (b) (c)

(d) (e) (f)

Fig. 3.4. Test images used (a) identical, (b) blocks of 4 pixels averaged, (c) gray scale, (d) inverted colors, (e) abstract mimic response, (f) blocks of 4 pixels shuffled.

$$p2 = \sum_{i \in A} \tau \times \delta(\neg(\exists \gamma, 1 \leq \gamma \leq k, i = u_\gamma)). \tag{3.6}$$

In this scheme, a larger p means lower quality. For our experiments, we use 6 images for A based on the guidelines suggested by [7]. These images are created by processing the query image with the following methods. (1) identical, (2) blocks of 4 pixels averaged, (3) gray scale, (4) inverted colors, (5) abstract mimic response, (6) blocks of 4 pixels shuffled. One set of test images used is shown in Figure 3.4.

3.2.3 Accuracy Metrics

Only the top 12 images ($k = 12$) are considered because they can fit in the screen of the PDA, as shown in Figure 3.2. The penalty τ is 13. Based on this accuracy

metric, we use the p with the full length of m_o features as the base. If m features are used, the accuracy is measured by the ratio of their values.

$$\text{accuracy} = \frac{p \text{ using } m_o \text{ features}}{p \text{ using } m \text{ features}}. \tag{3.7}$$

An example of our accuracy calculation is shown in Figures 3.5 and 3.6. Figure 3.5 shows the results using feature length m_o; (a), (b), (c), (d) and (e) are the desired results from the set A. The value of p1 is the sum of their ranks $1+2+3+4+5 = 15$ and p2 is 13. Thus $p_{m_o} = 15 + 13 = 28$. Similarly, from Figure 3.6 we obtain p1 to be $1 + 2 + 3 + 4 + 6 = 16$ and p2 to be 13, and $p_m = 16 + 13 = 29$. The accuracy ratio $= 28/29 = 96\%$.

We perform experiments by using different query images in various image collections containing 1000 to 10000 images. From our experiments, we find that for a given value of ρ, the number of features needed can be calculated using this formula:

$$m = 2400\rho^3 - 1800\rho^2 + 460\rho - 14 \tag{3.8}$$

to achieve at least 80% accuracy. As can be seen in Figure 3.7, for a database with similar images ($\rho = 0.4$), 30 more features are needed to achieve the accuracy when compared to a database with dissimilar images ($\rho = 0.03$).

3.2.4 Observations

The observations from the previous paragraphs can be summarized as follow: (1) The length of the features has a direct impact on the energy consumption. (2) The similarity index provides a quantitative measure between the query image and the images in the database. (3) Fewer features are needed to achieve the desired accuracy if the similarity index is low. Our system utilizes these observations to save energy. The features of the images in the database have been constructed before a query. When a query image is received, our system first calculates the complete n features

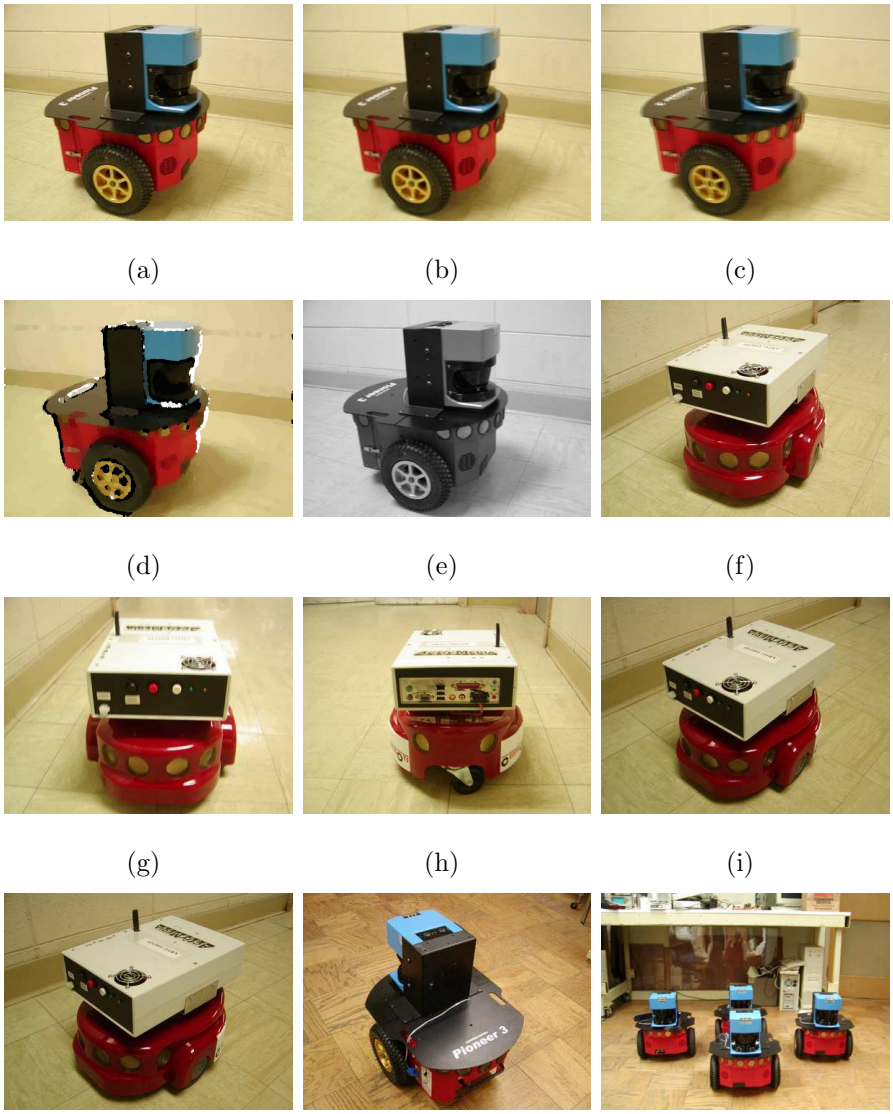


Fig. 3.5. Accuracy criterion for sample results using feature length m_o. The desired results are (a), (b), (c), (d), and (e) (from the set of test images introduced in the image collection). The other images are considered unwanted.

of the image and the similarity index ρ. The actual length m of the features is obtained by equation (3.8). The user has an option to add the query image into the database so that the new image is also compared in the next query. No additional

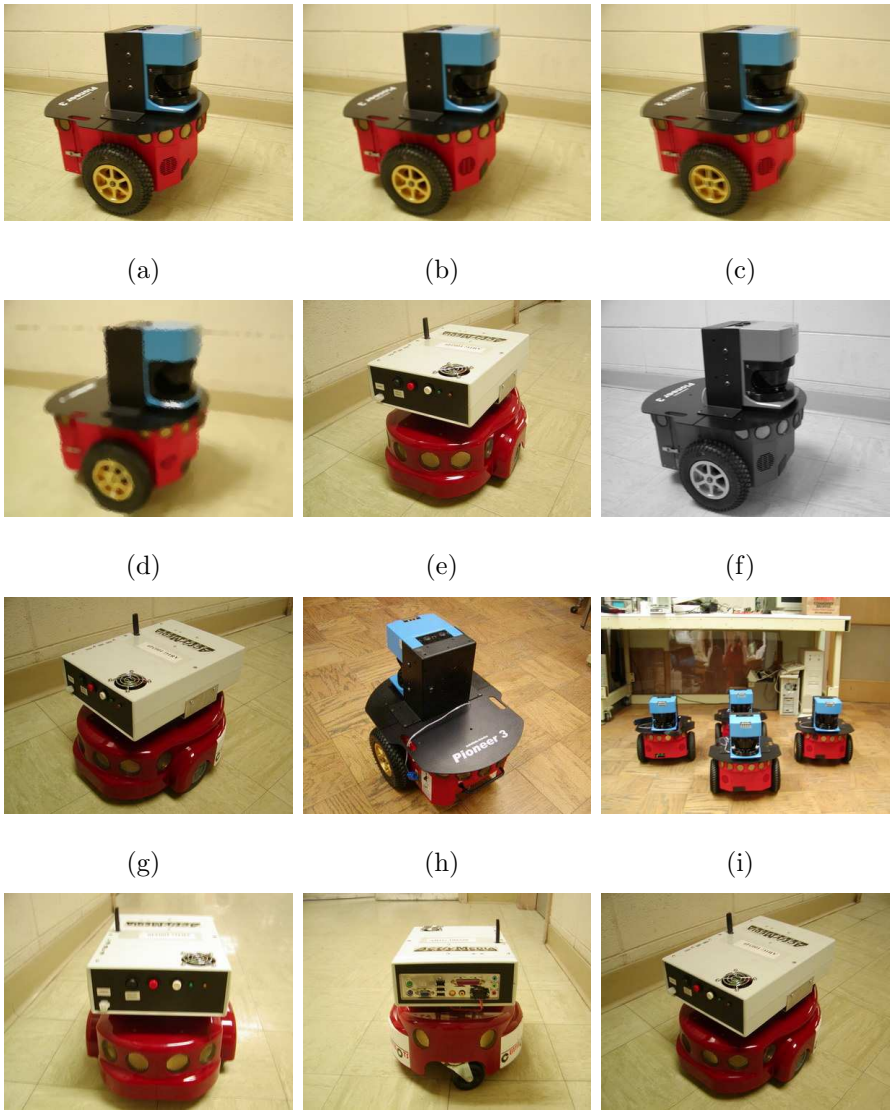


Fig. 3.6. Accuracy criterion for sample results using feature length $m = 20$. The images (a), (b), (c), (d), and (f) are the desired results. The orders of the unwanted images have no effect in calculating the accuracy.

computation is needed for the query image since its full-length features have already been computed.

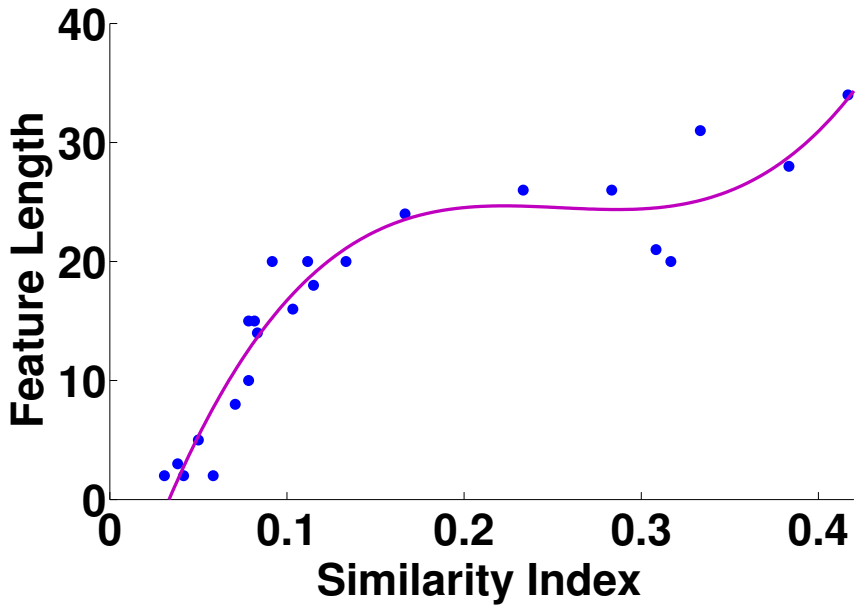


Fig. 3.7. The minimum lengths of features to achieve 80% accuracy for different similarity indexes.

The complete features of our image database are organized into 6 segments, each with 10 elements. Our system performs CBIR using the least number of segments needed, i.e. $10 \times \lceil \frac{m}{10} \rceil$. If the user is not satisfied with the search results, the user can raise the accuracy requirement to improve the search. An additional segment of features is used to improve the previous search results.

We can also estimate the amount of energy needed to search an image *before* performing CBIR (based on Figure 3.3). This information can be displayed on the screen so that the user can decide whether to perform this search. This is particularly useful when the database contains many images and a single search may completely drain the battery.

We measured the actual energy savings using databases of sizes 1000 and 5000 images. Figure 3.8 shows the energy savings for different query images for our algorithm compared to using a fixed number of coefficients ($m_o = 60$). For a database of 1000 images, the energy savings range from 23% to 38% (average: 34%). The mean

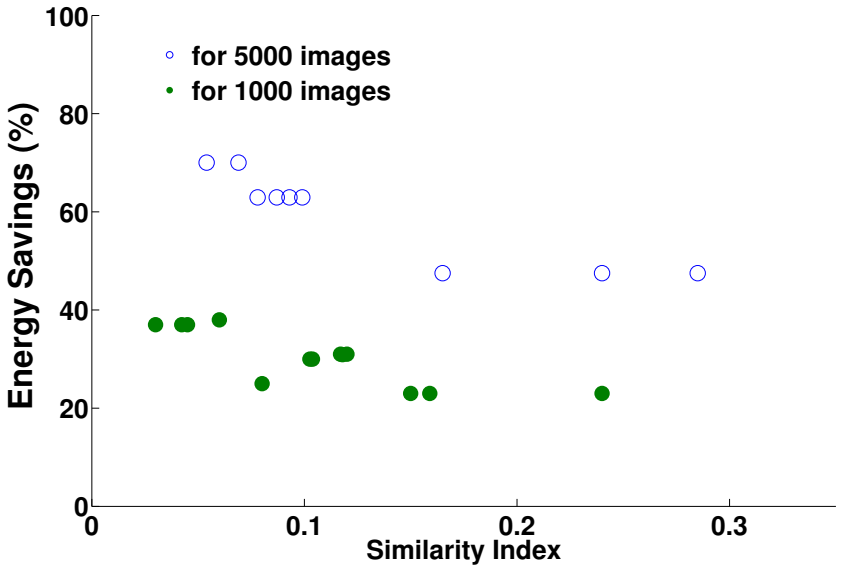


Fig. 3.8. Energy savings for different similarity indexes and 2 image databases.

value of m is 17. The accuracy range is from 78% to 100% with a mean of 90% and standard deviation 8.5%. For a database of 5000 images, the energy savings range from 47% to 69% (average: 59%). The mean value of m is 23. The accuracy range is from 74% to 100% with a mean of 85% and standard deviation of 8.7%. The overhead of calculating m in equation (3.8) is less than 2% of the search time for a database of 5000 images.

4. ADAPTIVE FEATURE LOADING

In this chapter we analyze the energy consumption for loading features from flash to memory. We compare the energy consumption of our interleaved scheme (introduced in Chapter 3) to the original implementation as the ratio of (3.3) and (3.2):

$$\frac{E_{int}}{E_{cbir}} = \frac{n \times (E_q + E_{d-a}(m_o, |I|) + E_s(m_o, |I|))}{E_d(|I|) + n \times (E_q + E_s(|I|))}. \tag{4.1}$$

A smaller ratio means less energy consumption by interleaving and thus more energy saving. In the following sections, we discuss 3 improvements to save energy: (1) selective loading, (2) adaptive loading, (3) database caching.

4.1 Selective Loading

In the first improvement, the feature length m is fixed equal to m_o. Equation (4.1) is simplified as

$$\frac{E_{int}}{E_{cbir}} = \frac{k + n \times E_{d-a}(|I|)}{k + E_d(|I|)}, \tag{4.2}$$

here $k = n \times (E_q + E_s(|I|))$. The value of n represents the number of queries and it signifies the energy reduction by loading only the features that are relevant to each query image. It indicates the number of *additional* searches to perform until this interleaving scheme consumes the same amount of energy as loading all the features at once before search.

```
SelectiveLoading(I, Q, m_o)
    ▷ For n queries { q_1, q_2 ... q_n } in query set Q
    for i ← 1 to n
          do
             ▷ Extract m_o features from query q_i
1            { f_1 f_2 f_3 ... f_{m_o} } ← q_i
             ▷ Load relevant features to memory
             for j ← 1 to m_o
                  do
2                    read I corresponding to f_j to memory

             ▷ Search for the query in image collection I
3            search q_i in I
```

As n increases, the energy consumed by the selective loading scheme will increase at a faster rate than the existing loading scheme. This is because for *each* additional query, the features relevant to that query need to be loaded into memory. In the existing scheme, all the features are loaded beforehand, and therefore need not be loaded for each query. It is to be noted that selective loading does not result in loss of accuracy as the features that are compared for the search remain the same as the existing loading scheme.

4.2 Adaptive Loading

All the schemes discussed so far involve loading a fixed number of features into memory. The existing scheme requires all features to be loaded into memory at once, and the selective loading scheme requires loading m_o features into memory each time. In the second improvement, we load a variable number of features; the number of features loaded depends on the difficulty of the query image. We have characterized the difficulty of the query using the similarity index defined in equation (3.4). Thus if the query is easier, fewer features need to be loaded into memory, and fewer features

are used for comparison. Fewer features are loaded into memory in two ways (1) The energy to load the features into memory $E_{d-a}(m, |I|)$ reduces adaptively with the number of features m (since $m \leq m_o$) (2) Each feature f_s for $s = 1$ to m has a smaller set of images in the inverted indexed feature database. This is because the segmented database mentioned in Section 3.2.4 is used. For example, when a feature length of 30 is used, it means that all the images in the collection are now represented by 30 features, and this results in a smaller feature database.

SelectiveAdaptiveLoading(I, Q, R, m_o)

▷ For n queries $\{ q_1, q_2 \ldots q_n \}$ in query set Q

for $i \leftarrow 1$ **to** n

do

▷ Extract m_o features from query q_i, denoted by f, $f_k \in f(i)$

1 $\{ f_1\ f_2\ f_3 \ldots f_{m_o} \} \leftarrow q_i$

▷ Compute Similarity Index ρ

2 $\rho \leftarrow \frac{\sum_{1 \leq k \leq m_o} |R(f_k)|}{m_o |I|}$

▷ Compute Adaptive Feature Length m

3 $m_a \leftarrow 2400\rho^3 - 1800\rho^2 + 460\rho - 14$

4 $m \leftarrow 10 \times \lceil \frac{m_a}{10} \rceil$

▷ Load m features to memory

for $j \leftarrow 1$ **to** m

do

5 read I corresponding to f_j to memory

▷ Search for the query in image collection I

6 search q_i in I

4.3 Handling Multiple Queries

We expect multiple queries in CBIR to be similar to each other. Images that are queried at around the same time are likely to be similar to each other, and this similarity is called temporal locality. Images that are taken around the same location

are said to exhibit spatial locality for the same reason. In the third improvement, we exploit both temporal and spatial locality for multiple queries to save energy. In some scenarios, a user may send multiple query images and they are similar. Figure 4.1 shows that a user walks towards a float in a parade and sends three query images. These images share some features (since the background and float are the same); this scenario is similar to keyword-based search when a user refines the search. For a query image, if some of the required features are already loaded into memory, these features will not be reloaded from flash again. Thus, the features residing in memory gradually increase with the number of queries. Eventually, the memory is full and some features must be evicted. We suggest using the LRU replacement policy but this situation does not occur in our experiments. We explore up to 5 successive queries in our experiments.

Let π_i denote the set of features for query image i. Let f_{ij} denote the j^{th} feature for query i, $1 \leq i \leq n$ and $1 \leq j \leq m$.

Now we assume that the previous $n-1$ queries have all their relevant features in memory. For the n^{th} query, the adaptive feature length is m_n. Now the required set of features is $\pi_n = \{f_{nj} | 1 \leq j \leq m_n\}$.

We thus include $|\pi_{n_1} = \pi_n \cap \{\pi_1 \cup \pi_2 ... \cup \pi_{n-1}\}|$ features directly from memory, and load the remaining $m_1 = |\pi_n - \pi_{n_1}|$ features from flash memory. We use $E_{db-a}(s, m_1)$ to denote the energy to load the database. Now the gain due to caching features is given by $\frac{E_{db-a}(s,m)}{E_{db-a}(s,m_1)}$.

SelectiveAdaptiveCachedLoading(I, Q, R, m_o)

▷ For n queries { q_1, q_2 ... q_n } in query set Q

for $i \leftarrow 1$ **to** n

do

▷ Extract m_o features from query q_i, denoted by f, $f_k \in f(i)$

1 { f_1 f_2 f_3 ... f_{m_o} }$\leftarrow q_i$

▷ Compute Similarity Index ρ.

2 $\rho \leftarrow \frac{\sum\limits_{1 \le k \le m_o} |R(f_k)|}{m_o |I|}$

▷ Compute Adaptive Feature Length m

3 $m_a \leftarrow 2400\rho^3 - 1800\rho^2 + 460\rho - 14$

4 $m \leftarrow 10 \times \lceil \frac{m_a}{10} \rceil$

▷ Required set of features: $\pi_i \leftarrow \{f_{ij} | 1 \le j \le m_i\}$

▷ Directly obtain cached features π_{i_1}

5 $|\pi_{i_1} = \pi_i \cap \{\pi_1 \cup \pi_2 ... \cup \pi_{i-1}\}|$

▷ Load m_1 features to memory

6 $m_1 = |\pi_i - \pi_{i_1}|$

for $j \leftarrow 1$ **to** m_1

do

7 read I corresponding to f_j to memory

▷ Search for the query in image collection I

8 search q_i in I

▷ Cache features in memory

9 cache π_i

(a) (b) (c)

Fig. 4.1. Multiple query images taken by a user as the user approaches a float. For the third query image, 35% of the required indexes are already loaded for the previous queries.

(a) (b) (c)

Fig. 4.2. Multiple query images taken by a user as the user approaches a building. For the third query image, 35% of the required indexes are already loaded for the previous queries.

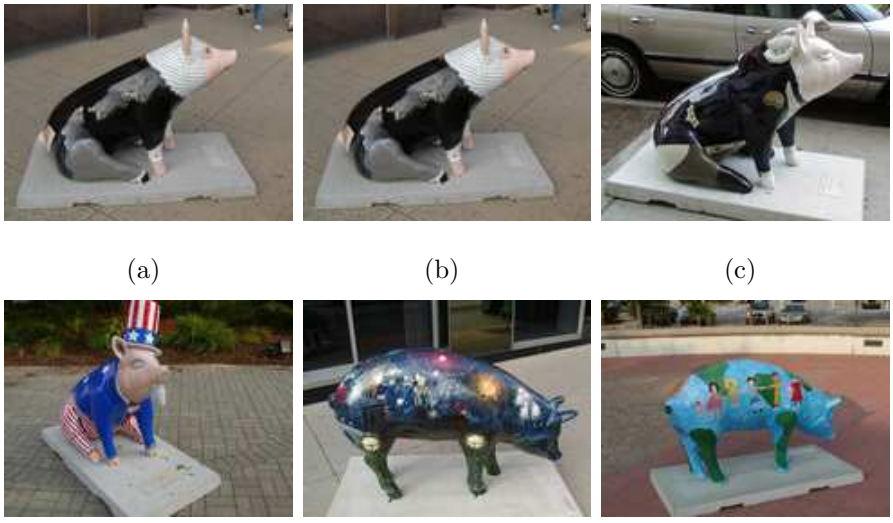
5. EXPERIMENTS

In this chapter, we present the experimental results obtained from performing CBIR for different image collections on the iPAQ. In Section 5.1 we provide timing measurements for CBIR . Section 5.2 describes the energy measurements. We measure the energy consumption of CBIR with and without the improvements proposed in Chapter 4, and we present the energy savings obtained.

5.1 Timing Measurements for CBIR

We perform CBIR using image collections of different sizes. Some sample results from our implementation of CBIR on the iPAQ are shown in Figure 5.1. The iPAQ has built-in 1.3-Megapixels camera and the average size of an image taken on the PDA is around 150 kB. If all images are taken by the PDA's camera, around 14000 images can be stored on the flash. However, the images used in our experiments are obtained by crawling the Internet because of the difficulty in obtaining such large image collections using the camera. We use up to 14000 images in our experiments. The average size of these images is 28 kB and they occupy around 400 MB in flash memory.

We measure the execution times for different image collections. As mentioned in Section 3.1, the execution time for CBIR can be decomposed into the database loading time t_d, feature extraction time t_q, and search time t_s.



(a) (b) (c)

(d) (e) (f)

Fig. 5.1. Results returned by CBIR. (a) is the query image in a personal collection of 1000 images and (b)-(f) are the top results returned.

5.1.1 Database Loading Time t_d:

Each image has the same number of features and hence the total size of the feature collection depends on the number of images. Loading the features from the flash memory t_d is a one time cost for the original implementation and does not grow with the number of queries. We measure the loading time for features of image collections up to 14000 images. We find that the time depends linearly on the size of the image collection for collections containing less than 10000 images. The relation we find in our experiments is: $t_d = 0.0065 * |I| + 1.2$. For collections containing more than 10000 images, we find that the iPAQ does not have sufficient program memory to accommodate all the features. We also measure the time taken to load features in our adaptive scheme given by t_{d-a}. We find in our experiments that t_{d-a} varies from 1-5 seconds depending on the feature length and the size of the image collection (upto 14000 images).

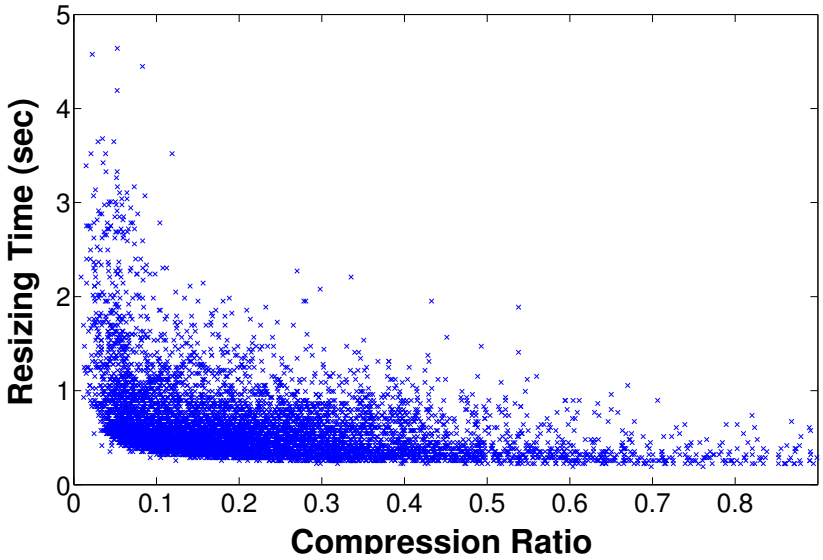


Fig. 5.2. Resizing time of images with different compression ratios. The compression ratio is the size of the JPEG image over that of the Bitmap format.

5.1.2 Feature Extraction Time t_q:

The feature extraction time includes the time to decode, resize the image, and then obtain the features from the resized image. Our experiments show that the time to obtain the features is a constant since every image has been resized. The time is 1.65 seconds. The decoding and resizing time is determined by the resolution and the compression ratio of the image. We show the resizing time versus the compression ratios and resolutions in Figure 5.2 and Figure 5.3. The compression ratio is defined as the size of the JPEG image over that in the Bitmap format. The figures show that the resolution is an important factor for predicting the resizing time.

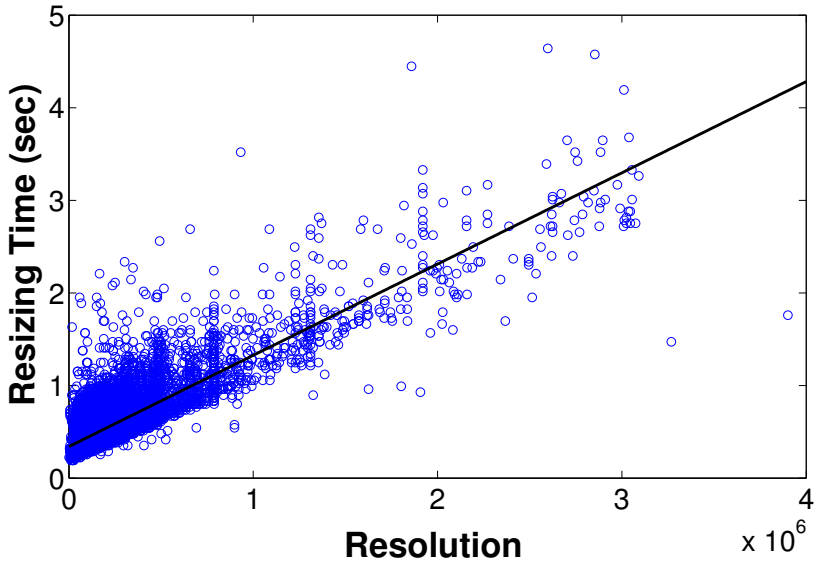


Fig. 5.3. Resizing time of images with different resolutions. The line shows the linear approximation of the data points.

5.1.3 Search Time t_s:

The search time depends on the size of the image collection and the features of the query image. As mentioned earlier, the database uses inverted indexing, so t_s is decided by the number of images in the database containing one of the m features of the query image. We show the search time for different query images under different sizes of image collections and we find that the average search time is 0.26 second for a collection of 10000 images. It is noted that the search time is relatively insignificant compared with t_d when the database contains the same number of images. For collections containing more than 10000 images, the search time is measured for the adaptive loading scheme and found to be linear to the size of the collection (we use collections upto 14000 images).

Fig. 5.4. Energy measurement setup for the HP iPAQ. Voltage samples are taken across a series resistor inserted between the iPAQ and its battery.

5.2 Energy Measurements for CBIR

We use the experimental setup shown in Figure 5.4 to measure the energy consumed by the iPAQ while performing CBIR. The figure shows the battery of the iPAQ connected to a data acquisition board. The current drawn by the PDA from the battery is obtained by measuring the voltage across a 0.25 Ω resistor. The voltage is sampled at 100 kHz using a National Instruments data acquisition card installed on a separate computer.

5.2.1 Energy Savings

We measure the energy for the following schemes: s1- existing implementation of ImgSeek, s2- selective loading, s3- selective adaptive loading, and s4- selective

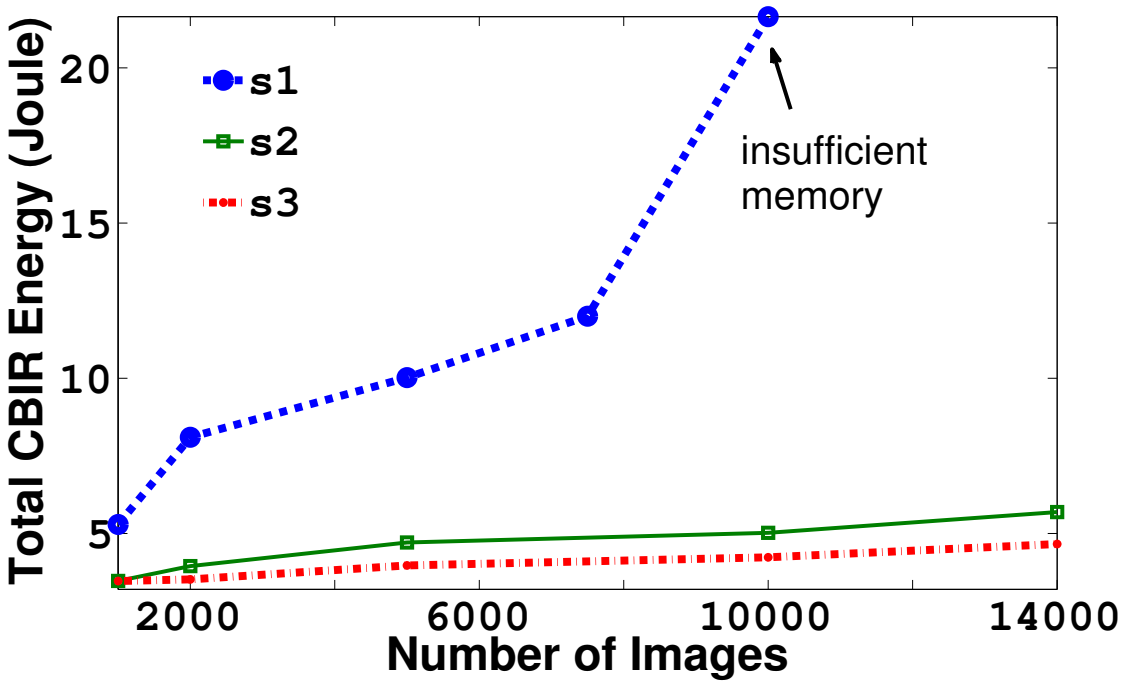


Fig. 5.5. Energy consumption of s1, s2 and s3 for mobile CBIR for a single query image for different sizes of image collections

adaptive loading with caching. It is noted that adaptive loading results in some loss of accuracy. In our experiments we observe that the accuracy remains above 80% for adaptive loading using the accuracy criterion defined in Section 3.2.3. Our measurements are performed for various query images. The query images are selected at random from the image collection stored on the flash. Six images are obtained from each query image by the modifications suggested in Section 3.2.3. We perform our experiments on image collections of different sizes from 1000 to 14000 images.

5.2.2 Energy for a Single Query

Figure 5.5 shows the total energy required to perform CBIR using s1, s2, and s3. Note that s4 saves energy only when handling multiple queries, so we show the

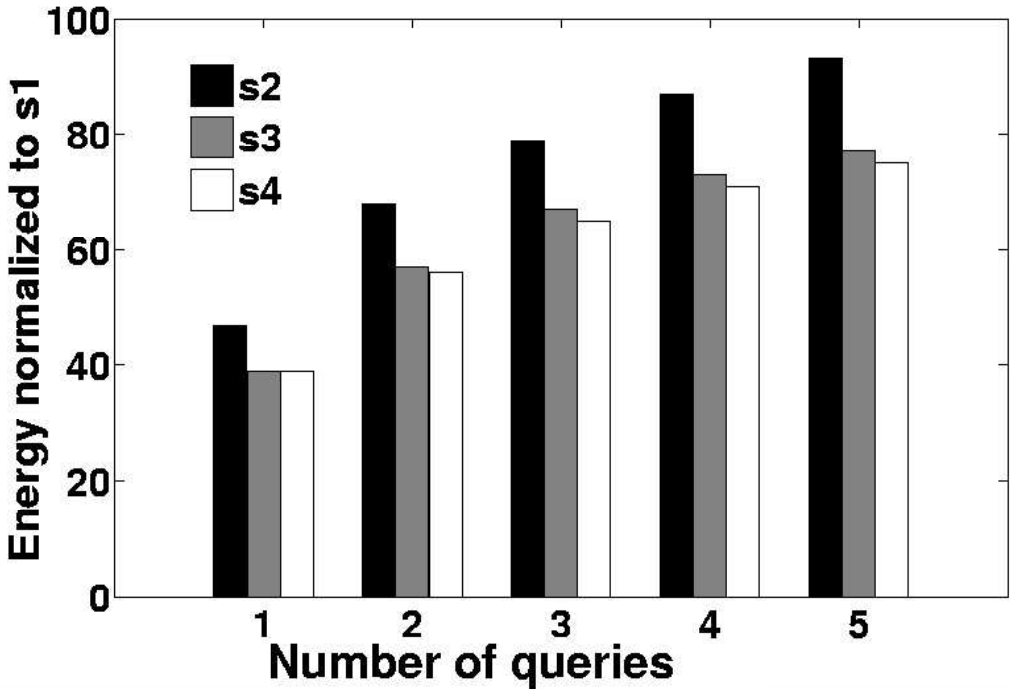


Fig. 5.6. Percentage energy consumption for mobile CBIR for different number of queries. Shorter bars mean more energy savings and are desirable.

results in the next subsection. For the existing implementation of ImgSeek, the PDA does not have sufficient memory to perform CBIR as the size of the image collection approaches 10000 images. The size of the feature index for 10000 images is 20 MB and we find that it does not fit in the program memory available in the HP iPAQ. With selective loading, the other schemes can be used for larger image collections. Both s2 and s3 demonstrate significant amounts of energy savings compared to s1. The schemes save 53 - 61.3% energy for the collection of 5000 images. This is the total system energy measured from the battery.

5.2.3 Energy for Multiple Queries

Figure 5.6 shows the energy consumption ratio $\frac{E_{int}}{E_{cbir}}$ defined in equation (4.1) for s2, s3, and s4 over s1 for a collection of 5000 images. When fewer features are in memory because of adaptive loading, energy savings span from 53% for a single query to 9% for five queries for s2. For s3, on average only 24 features are loaded into memory. The energy required to search also reduces because fewer features need to be compared. Additional energy is saved by s4 because even fewer features are loaded into memory as a result of caching. We observe 25% similarity on average between features of consecutive searches. For 24 features, this results in 18 features being loaded from flash, and 6 features reused in memory. The additional energy savings offered by s4 is small (0.08 J per query). This is because s3 already uses a small feature length. For s4, the features to be loaded into memory are further reduced due to caching. The overhead in accessing flash dominates and hence there is less room for additional savings. Moreover, we use random images for successive queries (both similar and dissimilar). Adaptive loading in s3 results in lower accuracy. If degraded accuracy is undesirable, s2 may be extended to include caching with the same number (m_o) of features.

As the number of queries increases, the energy consumption by s2 - s4 increases. For example the energy consumption by s3 (relative to s1) increases from 38.7% for a single query to 78% for five queries. In s1, all the features are loaded in memory at the beginning and no features need to be loaded between queries. In s2 - s4, a small subset of relevant features are loaded for each query. For a large number of queries, more energy is consumed to load the features for each query and the energy ratios increase. It may become more energy efficient to load all features into memory at once, but this is possible only if all the features can fit in memory. We extrapolate our measurements and find that s1 becomes more energy efficient than s2 after eight queries; and more efficient than s3 and s4 after more than 30 queries.

6. CONCLUSION

In this thesis, we present a CBIR system that performs search entirely on the mobile device. We present a loading method to save energy for performing CBIR on a PDA. The method adaptively selects the features to be loaded into memory, thus reducing the energy consumption while maintaining 80% accuracy. We also analyze the performance of CBIR over successive user queries, and show how features can be cached in memory to save energy. We implement our method on an HP iPAQ hw6945 PDA and show that we save 61.3% energy for a collection of 5000 images.

LIST OF REFERENCES

LIST OF REFERENCES

[1] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Image Retrieval: Ideas, Influences, and Trends of the New Age," *ACM Computing Surveys*, vol. 40, no. 2, p. 60, 2008.

[2] J. Forster, "Reducing Time and RAM Requirements in Content-Based Image Retrieval using Retrieval Filtering," in *Informatiktage*, pp. 143–146, 2007.

[3] O. Robles, J. Bosque, L. Pastor, and A. Rodriguez, "Performance Analysis of a CBIR System on Shared-memory Systems and Heterogeneous Clusters," in *Seventh International Workshop on Computer Architecture for Machine Perception*, pp. 309–314, 2005.

[4] A. Davidson, J. Anvik, and M. A. Nascimento, "Parallel Traversal of Signature Trees for Fast CBIR," in *ACM Workshops on Multimedia: Multimedia Information Retrieval*, pp. 6–9, 2001.

[5] S. Rudinac, G. Zajic, M. Ucumlic, M. Rudinac, and B. Reljin, "Comparison of CBIR Systems with Different Number of Feature Vector Components," *Second International Workshop on Semantic Media Adaptation and Personalization*, pp. 199–204, 2007.

[6] N. J. Gunther and G. Beretta, "A Benchmark for Image Retrieval using Distributed Systems over the Internet:BIRDS-I," in *SPIE - Internet Imaging II*, pp. 252–267, 2001.

[7] J. C. French, W. N. Martin, and J. V. S. Watson, "A Qualitative Examination of Content-Based Image Retrieval Behavior using Systematically Modified Test Images," in *Midwest Symposium on Circuits and Systems*, pp. 655–658, 2002.

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

[9] 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.

[10] T. Yeh, K. Tollmar, and T. Darrell, "Searching the Web with Mobile Images for Location Recognition," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 76–81, 2004.

[11] 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*, pp. 46–46, 2006.

[12] H. Sonobe, S. Takagi, and F. Yoshimoto, "Mobile Computing System for Fish Image Retrieval," in *International Workshop on Advanced Image Technology*, pp. 33–37, 2004.

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

[14] K. Kumar, Y. Nimmagadda, Y.-J. Hong, and Y.-H. Lu, "Energy Conservation by Adaptive Feature Loading for Mobile Content-based Image Retrieval," in *International Symposium on Low Power Electronics and Design*, 2008.