MOBILE IMAGE SEARCH USING MULTI-IMAGE QUERIES

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER ENGINEERING

By

Fatih Çalışır

August, 2015

MOBILE IMAGE SEARCH USING MULTI-IMAGE QUERIES
By Fatih Çalışır
August, 2015

ABSTRACT

MOBILE IMAGE SEARCH USING MULTI-IMAGE QUERIES

Fatih Çalışır
M.S. in Computer Engineering
Advisors: Prof. Dr. Özgür Ulusoy and Prof. Dr. Uğur Güdükbay
August, 2015

Visual search has evolved over the years, according to the demand of users. Single image query search systems are inadequate to represent a query object, because they are limited to a single view of the object. Therefore, multi image query search systems have gained importance to increase search performance.

We propose a mobile multi-image search system that makes use of local features and bag-of-visual-words ($BoVW$) approach. In order to represent the query object better, we combine multiple local features each describing a different aspect of the query image. Employing different features in search improves the performance of the image search system. We also increase the retrieval performance using multi-view query approach together with fusion methods. Using multi-view images provides more comprehensive representation of the query image. We also develop a new multi-view object image database ($MVOD$), with the aim of evaluating the performance impact of using multi-view database images. Multi-view database images from different views and distances increase the possibility to match the query images to database images. As a result, using multi-view database images increases the precision of our search system.

We compare our image search system with a state-of-the-art work in terms of average precision. In our experiments, we use single and multi image queries together with single viewed database. The results show that our image search system performs better with both single and multi image queries. We also performed experiments using $MVOD$ database and show that using a multi-view database increases the precision.

Keywords: Mobile visual search, content-based image search, query fusion, bag of visual words.

ÖZET

ÇOK GÖRÜNTÜLÜ SORGU YÖNTEMİYLE MOBİL GÖRÜNTÜ ARAMA

Fatih Çalışır
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanları: Prof. Dr. Özgür Ulusoy ve Prof. Dr. Uğur Güdükbay
Ağustos, 2015

Görüntü arama sistemleri kullanıcıların yönelimleri doğrultusunda yıllar içinde gelişim göstermiştir. Tek sorgu resmi kullanan resim arama sistemleri, sorgu nesnesinin sadece bir yönü hakkında bilgi sahibi olacağından, nesnenin tamamını temsil etme konusunda yetersiz kalmaktadır. Bu yüzden çoklu sorgu yöntemi önem kazanmaya başlamaktadır.

Sunduğumuz sistem, ilgi noktaları ve görsel kelime histogramlarını kullanan mobil bir çoklu görüntü arama sistemidir. Sorgu nesnesini daha iyi tanımlayabilmek için, her biri sorgu resminin farklı bir yönünü tanımlayan, çoklu ilgi noktası çıkarma yöntemi kullanılmış ve sonuçlar birleştirilerek kullanılmıştır. Çalışmamızda ayrıca farklı açılardan çekilmiş resimler birleştirilerek resim arama sonuçları iyileştirilmektedir. Farklı açılardan çekilen resimler sorgu resmini daha kapsamlı tanımlamaya olanak sağlamaktadır. Çalışmamızda ayrıca her nesnenin farklı açılardan ve uzaklıklardan resimlerinin bulunduğu bir veritabanı sunulmuştur. Veritabanının bu özelliği, sorgu resminin herhangi bir veritabanı resmi ile eşleşme ihtimalini artırmakta, sonuç olarak da resim arama sisteminin performansı artırmaktadır.

Resim arama sistemimiz literatürdeki önemli çalışmalardan birisi ile ortalama duyarlık açısından karşılaştırılmıştır. Bu deneyler sırasında hem tekli, hem de çoklu sorgu resimleri kullanılmıştır. Deneyler önerdiğimiz resim arama sisteminin daha iyi sonuçlar verdiğini göstermiştir. Ayrıca, yeni oluşturduğumuz veritabanı ile yapılan deneyler, farklı açılardan resimlere sahip nesnelerin veritabanında kullanılmasının resim arama sisteminin performansını iyileştirdiğini göstermiştir.

Anahtar sözcükler: Mobil görüntü araması, içerik temelli resim araması, sorgu birleştirme, görsel kelime histogramı.

Acknowledgement

I would like to thank Prof. Dr. Özgür Ulusoy and Prof. Dr. Uğur Güdükbay for their invaluable guidance throughout my research.

I am grateful to Assoc. Prof. Dr. Selim Aksoy and Asst. Prof. Dr. Muhammet Baştan for kindly accepting to be in my thesis jury and for their valuable comments and suggestions.

I would also like to thank to TÜBİTAK-BİDEB for their financial support with grant number 2228-A during my graduate education.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Scope

In this thesis, we propose a mobile content-based image search system that makes use of local features and Bag of Visual Words ($BoVW$) approach. In order to increase the performance of the search system we apply multi image query approach to represent the query images better. As depicted in Figure 1.1, multiple query photos are taken in our system by the mobile phone to create multi-view query. Then, the photos are sent to the server to be processed by the image search system to find the best matches.

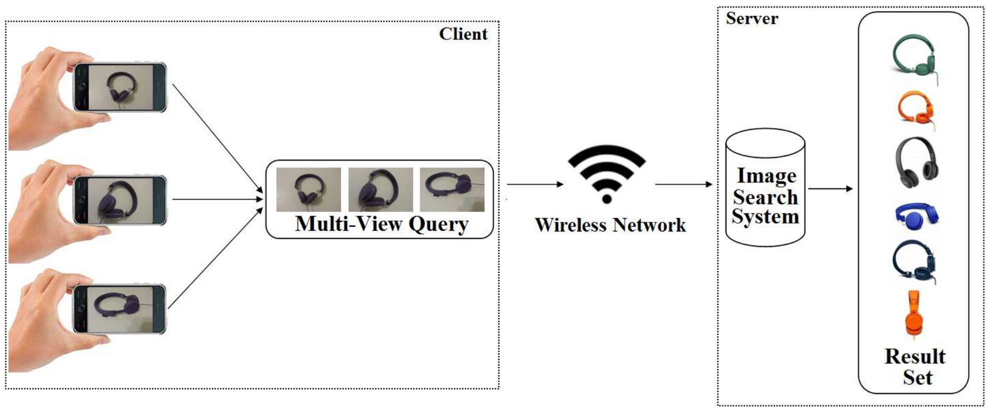


Figure 1.1: Workflow of our multi image query search system

We use *content-based* rather than *text-based* image search approach in our system. Although text-based search is much easier to formulate the queries, it suffers from *semantic gap* which is the inconsistency between the low level features and high level semantics of the query [3]. Moreover, the expressions used for queries in text-based systems can be ambiguous and longer. On the other hand, the content-based image search systems use the local features extracted from the query images. These features are reproducible when the same extraction method is used, which makes them unambiguous.

Features extracted in a content-based image search system represent the important characteristics of the image that help identify and differentiate that image from the others. The representation of an image becomes more accurate when it has more features. Although there are different methods to extract features from images, there is a limit when focusing on a single image. Even the best feature extraction method has a representation limit. In order to extend the limits of representation, multi query approach is adopted. The same object is represented as a combination of multiple images of its different views and scales. Not only one side or scale of an object, but also the other parts of it is included in the representation process. In this way, the representation becomes more comprehensive and accurate.

The proposed mobile image search system enables the user to apply multi query approach through a simple-to-use platform. Taking photos from different angles and distances is very easy with mobile phones. Moreover, the accessibility to the search system is very high because mobile phones have become an essential part of our life. With the recent advances in Internet technology, everybody carries a potential image search system in their pockets.

In the literature, various methods are provided to extract local features. In order to represent the images better, we combine the multiple local feature methods. Then, the combined local features are used in the $BoVW$ approach. The local features are clustered in order to create visual words that form the visual vocabulary. Using this vocabulary, histograms are constructed to represent the images.

Because of the speed and bandwidth limitations on the mobile Internet connection, instead of the whole image, the mobile client extracts the local features and sends them to the server. The histogram of the query image is constructed on the server. The query histogram is then compared with the other histograms stored in the database, and similar images are sent to the mobile client as the results of the image search. During the matching process, the similarity between query and database histograms is calculated. We use various similarity functions and analyze their performance.

We also use multi image query approach in our system, considering that using single image query results may not contain adequate information about the query object. It only allows to gather partial information from the query object, because the single image query is not able to cover all the aspects of a query object. It only focuses on a single view of the object. For this reason, we employ a multi image query approach to increase the amount of information about the query object and obtain better results [4].

The multi image query approach requires to combine the queries or the result lists obtained from queries [5]. The combination methods are categorized into two, according to the combination approach. If the query histograms are combined, then the method is called *early fusion* because histograms are combined before they go into the search process. In *late fusion*, the query histograms are processed by the image search system. After the results of the query histograms are received, they are combined into a single result list. We do experiments with both types of combination methods and evaluate their performance.

In our work, we also prepared and used a multi-view database in which each object has at least two images taken from different views. With this approach, our aim was to increase the amount of information about the objects stored in the database. We performed experiments by using different combinations of queries (single and multi-viewed queries) and database (single and multi-viewed database).

In summary, we propose a mobile application that performs content-based image search by using multi image query approach. In order to improve the performance of the system, we use multiple feature extraction methods, early and late fusion methods, and propose multi-viewed database.

1.2 Contributions

The main contributions of this thesis are given as follows.

- We propose a multi-image mobile visual search system which allows user to take multiple photos of an object and search for similar ones stored in the database.
- We provide a detailed analysis on the comparison between single and multi-image query processing approaches in terms of the precision of the system. We also investigate the performance impact of single and multi view databases.
- We analyze the performance of different similarity functions used to match query and database images. Additionally, we also analyze the performance of different fusion methods used to combine queries or the result lists.
- We provide a new multi-view image database ($MVOD$) in which objects have multiple images representing different views and scales. We also compare the running times of single and multi-view queries.
- We have developed a mobile application that involves a multi image query processing approach and parallelizes the process to reduce the execution time. After taking a query photo, while the user is taking the next one, the features of the previously taken photo are extracted on the mobile device and sent to the server. On the server side, these features are processed and the results are stored to be used together with other query images, if any.

1.3 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 summarizes the works related to content-based image search, mobile visual search, single and multi query approaches. Chapter 3 provides the details of the image search system with a single image query approach. Chapter 4 describes the multi image query search system together with the fusion methods and the mobile side of the system. Chapter 5 explains the test environment, databases, evaluation methods and the experimental results. Finally, Chapter 6 concludes with the discussion of the results and provides some possible directions for future research.

Chapter 2

Related Work

2.1 Content Based Image Retrieval (CBIR) Systems

In recent years, image retrieval systems have gained importance in parallel with the advances in technology, which has resulted in a huge increase in the number of digital images, multimedia files, and visual objects. Digital images are used in various fields like engineering, science, medicine and architecture, and this makes both the retrieval and processing mechanisms of large image databases important. There are two basic types of image retrieval methods: *text based* and *content based*. Text-based image search is not based on the image content, but it uses the annotations assigned manually [6]. Text-based image retrieval systems involve *query-by-keyword* paradigm that makes use of keywords or sentences to formulate queries [3]. Such image retrieval systems have various limitations. Annotations require a considerable amount of human labor and time, which results in ambiguous content [6, 7, 8]. Humans have different intuition and this may lead to different annotations for the same images by different people. Due to such drawbacks and limitations, annotated image databases are not widespread [2]. As a result, content-based image retrieval (*CBIR*) has become popular and attracted attention. *CBIR* systems involve *query-by-example* paradigm that make use of examples or sketches while formulating the queries [3]. In *CBIR* systems, relevant images are retrieved from an image database by using image features derived automatically from the image itself [8].

2.1.1 Example Systems

There are many image retrieval systems developed by incorporations or research labs of universities [9]. *IBM QBIC* system, developed by IBM Almaden Research Center, is an image retrieval system that allows different types of queries. User can use example images as queries, draw some sketches or use color or texture patterns. For color option, for example, the user can draw magenta circle on green screen. For texture option, users need to select texture patterns from predefined texture images [10]. Another system is *Photobook* system developed by the MIT Media Lab. This system uses both text annotations and content of the query images. Text annotations are used to narrow down the possibilities in the database like selecting cloth samples for curtains. Additionally, this system offers relevance feedback mechanism that allows improving the ranking result, according to the choice of the user [11]. Another image retrieval system is *VisualSeek*, which is developed at Columbia University. This system divides the query into regions automatically and then uses color and spatial information while retrieving the relevant images [12]. *WBIIS* System, developed at Stanford University, provides a partial sketch searching and characterizes color variations. It uses a wavelet-based method [13] to index images. Another CBIR system, called *Blobworld*, is developed at the University of California-Berkeley [14]. Blobword divides the images into regions (*blobs*), which are assumed to correspond roughly to the objects. A few of these blobs are then used for querying, which are described by using color distribution and texture descriptors. Lampert describes a similar system with Blobworld, which finds the local regions that best match a query [15]. The background of the image is then cluttered. *Branch and bound* object localization method is used to find the local regions, which probably are the objects. *Sketch4Match* [16] is another CBIR system that uses sketches drawn by user. Edge Histogram Descriptor (*EHD*), Histogram of Oriented Gradients (*HOG*) and Scale Invariant Feature Transform (*SIFT*) are used as features. Another recent work [17] presents a system that uses *Windows Azure* cloud platform for parallelization. The system extracts both color and texture features from the images, and *Euclidean Distance* is used to compare them. In addition to these systems, Liu et al. [18] propose an efficient image search system which combines different descriptors (*histogram of gradients, local binary pattern* and *color histogram*) which are extracted from a single query. Then, these descriptors are mapped to a histogram using proposed *multiview alignment hashing* method.

2.1.2 CBIR Implementation

In CBIR systems, there are three main steps: (i) extracting and storing features of database images, (ii) extracting features of query images, and (iii) matching and ranking [7]. The general workflow of a CBIR system is shown in Figure 2.1.

2.1.2.1 Feature Extraction

Features are essential components of an image that describe it for CBIR approaches. The overall performance of the system is as good as its feature detector [6]. So, a good feature extraction method is needed and global features are not adequate to describe the image completely. The precision of the CBIR systems that use global features are quite low due to the semantic gap, which is the difference between actual features of an object and how it is represented [19]. The most commonly used global feature is color, and it is generally represented as color

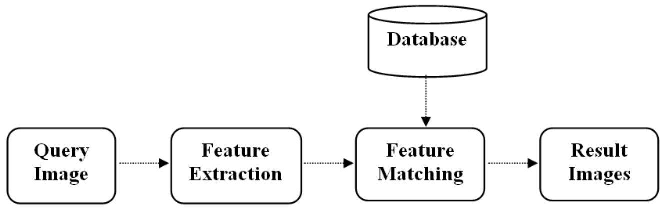


Figure 2.1: General workflow of a CBIR system

histogram. For example, assume there is an image that has two flowers in red and yellow. When we use the global average color feature, we get orange color, which is not a correct representation of the initial image [20]. Therefore, local features are a better choice to represent an image and almost all CBIR systems use local features. While using local features, both interest region (*key point*) detectors and descriptors are needed. Girod et al. [21] and Mikolajczyk and Schmid [22] state that independent of the key point detector used, *SIFT* outperforms all the other descriptors and remains as the most popular one. However, making the same kind of generalization for the key point detector is not possible.

There are various feature detectors in the literature. Mikolajczyk and Tuytelaars summarize them in their work [23]. *Harris* is a corner and *Hessian* is a blob detector. Both of them are rotation invariant. *Difference of Gaussians (DoG)* is another feature detector that detects both corners and blobs, and it is rotation and scale invariant. The properties of the *SURF* detector are similar to *DoG*, and it is more efficient in terms of calculation time [23]. In addition to these detectors, there are *Harris-Affine* and *Hessian-Affine*, which are corner and blob detectors providing rotation, scale and affine invariance.

Shen et al. [2] state that choice of techniques, parameters and threshold values may vary according to the database used and the application domain. The choice of key point detector is included in this category. However, there are some research that compares the key point detectors. Two of them become prominent among the others, these are *HARRIS*, which is a corner detector, and *SURF*, which is a blob detector [23, 1, 21, 24].

2.1.2.2 Retrieval

In CBIR systems, similarity matching is used rather than exact matches because the intent of the user is not to find the same but similar images to the query image. In order to perform this operation, the extracted features must be compared. The retrieval is performed with *BoVW* approach, inspired from the *Bag of Words (BoW)* representation used in text retrieval [25]. The images are treated as a document that contains visual words. Then, the retrieval process is similar to text retrieval. The images are represented as the set of visual words and the comparison is performed between these sets.

2.2 Mobile Visual Search

Rapid development of hardware and software technologies has turned the mobile phones into powerful image and video processing devices equipped with high-resolution cameras and Internet connection. This property of the mobile phones makes mobile visual search systems popular. Although the mobile search systems have some advantages like ease of use and accessibility, this new platform has its unique challenges [1]. Computation power, storage, battery and network bandwidth are all limited [24]. Therefore, some client-server architecture models were introduced, as shown in Figure 2.2.

In the first architecture depicted at the top of Figure 2.2, the query image is sent to server without any processing. The feature extraction and matching are performed on the server side. Then, the obtained result list is sent back to the mobile client and displayed to the user. The second architecture extracts the features of the query image on the mobile side, and sends the features instead of the whole image to the server. When the features are received on the server side, matching is performed, and the results are sent to the mobile client as shown in the middle of Figure 2.2. The last architecture, displayed at the bottom of Figure 2.2, perform feature extraction and matching on the mobile client using a local database. If there is a match, then the results are presented to the user. However, when there is no match, the features are sent to the server, and matching is performed using a large database.

Image databases require large storage space, therefore storing the database on a mobile phone is not a good solution for most of the visual search systems. When the database is stored on the server side, the query image has to be sent to the server side to be compared with the images in the database. However, sending the

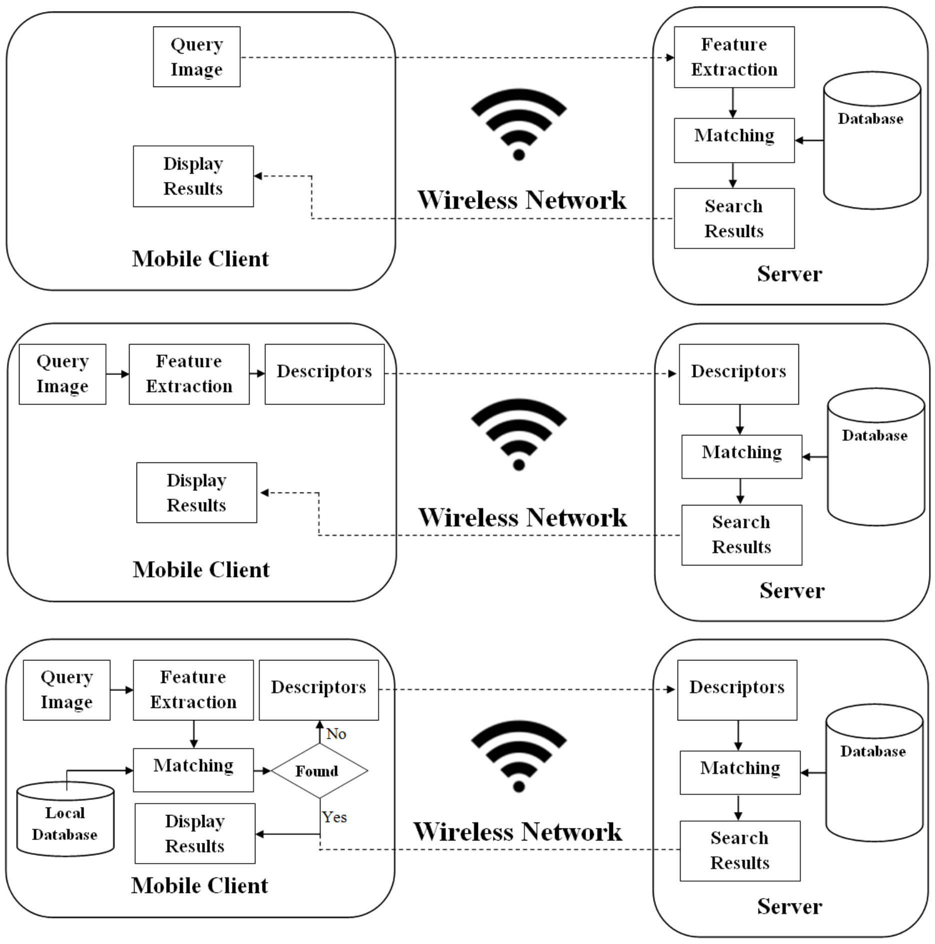


Figure 2.2: Client server architectures for mobile visual search systems [1]

whole image is time consuming due to the bandwidth of the Internet (especially using *3G* technology) and the size of the image. So, extracting features of the image at the mobile side and sending these extracted features to the server side is the best choice if the search system has a large database [1, 21].

2.2.1 Example Systems

Mobile visual search is a promising field so there are lots of work focusing on this area. There are excellent surveys on mobile visual search that examine different choices about client-server architectures, key point detectors, descriptors and types of queries [1, 21, 24, 26, 27]. All of these works use common $BoVW$ approach.

Basically, they extract key points and descriptors, create visual words and find similar images from the database. Griod et al. [1, 21] experiment with different client-server architectures by sending images to the server, sending features to the server and sending features to the server progressively with wireless and *3G* connection. They find out that if the phone uses wireless connection and the connection is *3G*, then sending features to the server is better than sending the whole image in terms of response time. Additionally, if the features are sent progressively, meaning while extracting features at the same time sending them to the server, it further reduces the response time. These works also argue that *SIFT*, *DOG* and *HARRIS*, which are affine detectors, are slow to compute but highly repeatable. *FAST* is a fast key point detector, however, it has low repeatability. As a feature descriptor, *SIFT* outperforms all the others independent of the used key point detector. Su et al. [24] focus on multimodal queries for mobile visual search systems. They combine results of multiple key detectors applied to the same image and they conclude that combining features results in an increase in the accuracy of the search system. Chandrasekhar et al. [26] and Duan et al. [27] further focus on the local and compact descriptors for mobile search systems. Quantization is needed to compress the local descriptors in order to increase the response time of the system.

There are also many research works that apply image retrieval methods to mobile image search. Hare and Lewis [28] propose a system that works on exhibit images by using the vector space model. Noda et al. [29] and Sonobe et al. [30] describe domain specific systems that focus on fishes and flowers, respectively. Shape and color features are used to distinguish the objects (fishes and flowers). Yeh et al. [31] describe a system that finds the source pages of relevant images on the Web. The system requires two photos: one with the object and the other one is the same background but without the object. The object is then extracted from the difference of two images. Girod et al. [21] propose an image categorization system that uses the Difference of Gaussians (*DoG*) as the feature detector and *SIFT* for the feature descriptor. The features are indexed with Growing Cell Structure (*GCS*) which is an unsupervised clustering method. For the quantization part, nearest neighbor clustering with the normalized *L2* distance is used. The similarity comparison is done with an inverted *L2* similarity metric. For the categorization process, top n images are found and the category that has the highest number of images is assigned as the category of the query.

Chen and Girod [32] describe a mobile product recognition system where the products are CDs, DVDs and books that have printed labels. The system is local feature based, and Compressed Histogram of Gradients (*CHOG*) and *SIFT* are used for local feature extraction. In their work, two client server architectures, which are sending features and sending images, are implemented and compared in terms of response time. While sending feature mode requires five seconds, sending image mode needs ten seconds to respond. Joint Search with Image Speech and Words (*JIGSAW*) [33] is another mobile visual search system that provides multimodal queries. This system allows user to speak a sentence and performs text-based image search. The user selects one or more images from the resulting set to construct the visual query and the constructed multimodal query is used for content based image search. The system uses both local features (*SIFT*) and color histogram. In [15], a mobile product image search system that automatically extracts the object in the query image is proposed. From the top n images that have a clean background, object masks are found. The object in the query image is then extracted by using a weighted mask approach and its background is cleared. The cleaned query image is finally used to perform image search.

Girod et al. [1] describe a mobile visual search system that adopts the client-server architecture where the database is stored on the phone. The system has a common *BoVW* approach, and 4 different compact database methods are tried and their performances are compared. Li et al. [34] propose another on-device mobile visual search system. The search system uses *BoVW* approach with a small visual dictionary due to the memory limitation. Additionally, the most useful visual words are selected to decrease the retrieval time considering the processor limitation. Guan et al. [35] describes another on-device mobile image search system which is based on bag-of-features(*BOF*) approach. The system uses approximate nearest neighbor to use high dimensional *BOF* descriptors on the mobile device with less memory usage. The search system also utilize the *GPS* data from the mobile device to reduce the number of images to be compared.

Mennesson et al. [36] propose a method that uses *elementary blocks* which are well-chosen subset of the local features. This approach decreases the size of the data which is sent to the server side for image search. Another mobile visual search system is proposed by Ji et al. [37]. The search system uses *3D point cloud* to consider the depth information from the query image. Then, sparse coding is used to construct the *BoW* histograms. Zhou et al. [38] describes a mobile image search system which does not use a codebook which decreases the used memory and quantization errors. The system makes use of *SIFT* descriptors, and the dimension of the descriptor is decreased using *PCA* method. The descriptors whose dimensions are reduced are called *PSIFT*, and approximated nearest neighbor search is applied to the them.

In addition to these works, there are various commercial mobile visual search systems. *Google Googles* [39] is a mobile search system that allows the user to take a photo or upload previously taken photo and use it as a query image. *oMoby* [40] is another mobile visual search application that performs a general object search. The information about the taken image can be seen by the user on the screen. Another application is *Point&Find* [41] which is developed by Nokia. This system does not require taking photos. The users point the camera to the scene or object and get the corresponding information about it. *Kooaba* is another mobile visual search application developed by Kooaba. It is a domain specific search system and target domains are books, DVD and game covers [42]. *iCandy* [43] is developed by Ricoh; it allows users to search media covers. Another mobile search system, *Snap2tell* [44], is developed by Amazon. This system follows general mobile search system methods and user takes a photo and gets relevant information. *Digimarc Discover* [45], developed by Digimarc. Inc., is similar to *Point&Find* application; the user points the camera to the object and gets the information. Media Cover Recognition [46] is a domain specific mobile search system developed by Stanford University. It focuses on the media (CD, DVD) covers. *PlinkArt* [47] is another domain specific mobile search system developed by Google, whose target domain is well known artworks. The user takes a photo of a well known artwork and gets corresponding information about it. One of the latest mobile search application is *CamFind* [48] developed by Image Searcher Inc. It is a general object search system. When the user takes a photo of a scene, products are identified and similar objects are listed as a result. Salai proposes a mobile image search system that is developed on *Android* platform [49], and it uses color histograms extracted from *HSV* image. The other recent mobile search application is *Flow Powered by Amazon* which is developed by Amazon [50]. It requires to point the camera to the object, and the application scans it and delivers the related information to the screen. Another application is *TapTell* [51] which performs interactive visual search. The user indicates the object with drawing circular shapes, then the application provides conceptual recommendation according to the result of the image search.

2.2.2 Multiple Image Query Approaches

Using a single image as a query may not be expressive enough. A single image can only show one aspect of an object such as the front of a car but not the rear. If there is an object occlusion in the image, the expressive power of the query image decreases further. Using multiple images as queries allows covering different viewpoints, which increases the expressiveness [52, 53]. Two different approaches are presented in the literature about the multiple image query. The first approach (*early fusion*) is combining queries before the search and then using the combined query in the image search. In the second approach (*late fusion*), the queries are performed individually and the retrieved results are combined.

Several methods for multiple image querying are described in [54, 55]. *Joint AVG* is a method for early fusion. Query histograms are combined into single histogram by taking average values. *Max AVG* is similar to *Joint AVG* and the only difference is selecting the maximum value instead of average value. *MQ AVG* is a late fusion method. The lists returned by each single query image are combined into a single list with taking average scores. *MQ Max* is similar to *MQ AVG* and the only difference is instead of average scores, the maximum score is used in this method. Mazloom et al. [55] state that the methods in which the average operation is used can be preferred because the average operation is more robust to noise.

There are various works that focus on multiple image querying. Mazloom et al. [55] describe an event classifier system that performs the classification on videos, but its working principle is the same as the multiple image querying. From videos, a few example screenshots are extracted and the rest of the system performs an image search using multiple image queries. Both early and late fusion methods (average and maximum methods) are applied and average methods are preferred because of their robustness to the noise. Arandjelovic and Zisserman [52] propose an object retrieval system that applies multiple queries approach. In the system, the user first enters a textual query and using this textual query, Google image search is performed. The top eight images are then retrieved and each of these images is treated as query images. Early and late fusion methods are applied. Tang and Acton [56] propose a system that extracts different features from different query images. These extracted features are then combined and used as the features of the final query image. Another system is proposed in [57], which allows users to select a different region of interest. Then each region is treated as queries and results are combined. Zhang et al. [58] describe a similar system, which also uses regions; however, these regions are extracted automatically and users select regions from the extracted parts. Joseph and Balakrishan [8] propose a system that uses multiple queries and local binary pattern (LBP) texture descriptors. Then, the logical AND operation is applied, which gives an image that is similar to both query images. With this approach, nested operations are possible. Xue et al. [59] propose a system that uses multiple queries to reduce the distracting features by using a hierarchical vocabulary tree. The system focuses on the parts that are common in all the query images. In [54], another multi query system that uses early fusion is described. In this system, each database image is compared with each query image and each query image gets a weight according to the similarity between the query image and the database image. The weights of query images change according to the database image. The system proposed by Fernando and Tuytelaars [53] uses an unsupervised pattern mining method in order to find the object of the interest exactly. Using multiple query images, it extracts mid-level representations, which are the regularity in the query images.

2.2.3 Proposed Mobile Visual Search System (BilMobile-IM)

In this section, we describe the basic characteristics of our image search system, *BilMobile-IM*, as compared to the previous works. Most of the previous image search systems use local features instead of global features, because local features are invariant to scale and rotation. Moreover, they perform well when there is occlusion in the image [60]. Considering these benefits, we also use local features in our image search system. Instead of using a single local feature detector, we use two different key point detectors and combine their results. With this method, the amount of information gathered from the query images increases. In accordance with this, the overall performance of the system also increases.

We also use a multi image query approach in our image search system. Using multiple images of the query object taken from different views allows us to gather more information about the object. While we observe the query object from a single view in a single image query approach, it is possible to increase the amount of information by taking multiple photos from different views. The important part of the multi image query approach is the fusion of the information gathered from different query images. We analyze various fusion methods existing in the literature and compare their performance.

We provide a new database where the objects have multiple images that represent different views of the objects. While using the database, we also adapt two different similarity methods to compare query and database images. Increasing the amount of information in the database images increases the performance of our search system.

While applying different methods in the matching part, we also analyze the time spent by these methods. We examine the matching times of the methods to provide information about the practicality of the similarity methods.

Chapter 3

Image Search System

The single-image query approach that we describe here forms the basis for our multi-query image search system. The general architecture of the single image query approach is shown in Figure 3.1.

First, features of the query images are extracted and the vocabulary is created accordingly. Then, the histogram of each query image is created by using the vocabulary, and the created histograms are stored in the database. When a query image is received by the search system, its features are extracted and histogram is created from these features. Finally, the query histogram is compared with the histograms stored in the database. The best k results are returned as the result list of the image search system. The following sections describe the single query approach in detail.

3.1 Feature Extraction

The key elements of our system are the features extracted from the images. They are the pieces used to create the description of the images. In order to have a good representation, we need to detect the "key" parts of the images. For that purpose, we have used key point detector and descriptor methods.

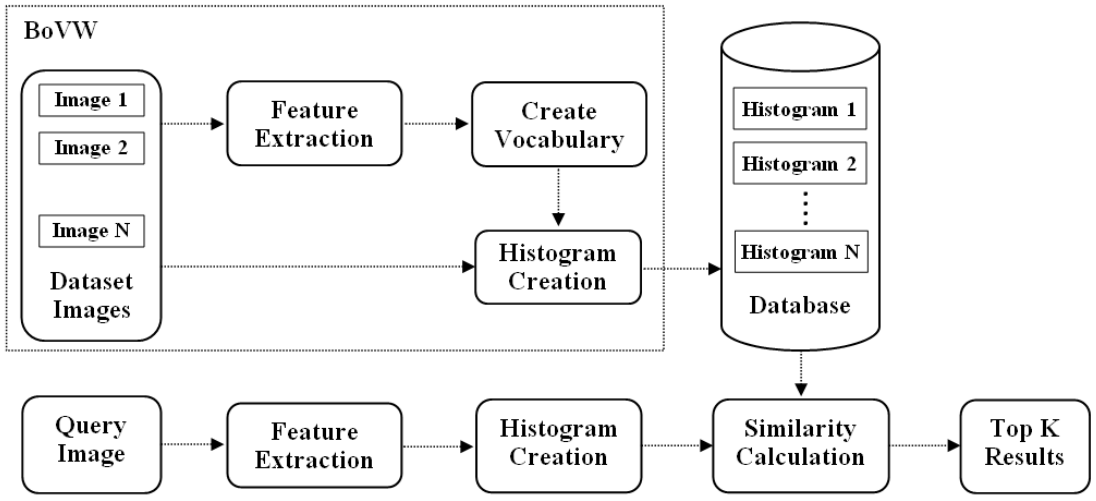


Figure 3.1: General system architecture of a single image search system

Using good key points (*local features*) has various advantages. First of all, they are repeatable, which means under different viewing or scale conditions, same features can be found in the image. The other property is the locality, which makes the local features robust against the occlusions. They are also informative, which allow to distinguish and match with each other [61]. These properties make local features preferable for image search systems.

We use *HARRIS*, which is a corner detector, and *SURF*, which is a blob detector, key point detectors. The detected *HARRIS* and *SURF* key points are shown on a sample query image in Figure 3.2.

The client-server architecture of our system requires calculating key points on the mobile device which makes the speed important in terms of execution time and user satisfaction. Although *SIFT* key point detector performs well, it is comparably slower than *HARRIS* and *SURF*. We select *HARRIS* and *SURF* key points because of their speed.

The other reason to select these local features is that, they have the properties which a good local feature needs to have. *HARRIS* local feature is robust against blurred images, scale, viewpoint and illumination changes [61]. Similarly, *SURF* local feature is invariant to scale and view point changes. These local features give the best performance results in our image search system.

3.1.1 HARRIS Key Point Detector

The corners are one of the important local features in images because they are the points which have high curvature, are located at the junction of different brightness regions, are not affected by illumination and have rotational invariance [62]. *HARRIS* is a corner detector that has all these properties. It is an intensity-based corner detection method; it locates the corners directly from the grayscale values. This property makes *HARRIS* fast and independent from other local features [62]. In Figure 3.2, we can observe the *HARRIS* key points on the left and they are located on the corner of the image.

3.1.2 SURF Key Point

The *SURF* key point uses *Hessian* matrix and detects blob-like structures where the determinant is maximum [63]. *SURF* can be interpreted as the fast version of *SIFT* and it achieves that by using *integral images*. *SURF* uses square-shaped filters as an alternative to Gaussian smoothing. The combination of square filters and integral images makes *SURF* faster because calculating the sum of intensities inside the square filter takes *O(1)* time [63].

In Figure 3.2 we can see the *SURF* key points on the right. It is a bit harder than *HARRIS* key points to observe the pattern, but the key points are located at the edges where the color (*gradient*) changes.

3.2 Bag of Visual Words (BoVW) Approach

BoVW approach is inspired from the *Bag of Words (BoW)* representation used in text retrieval [25]. The images are treated as a document that contains set

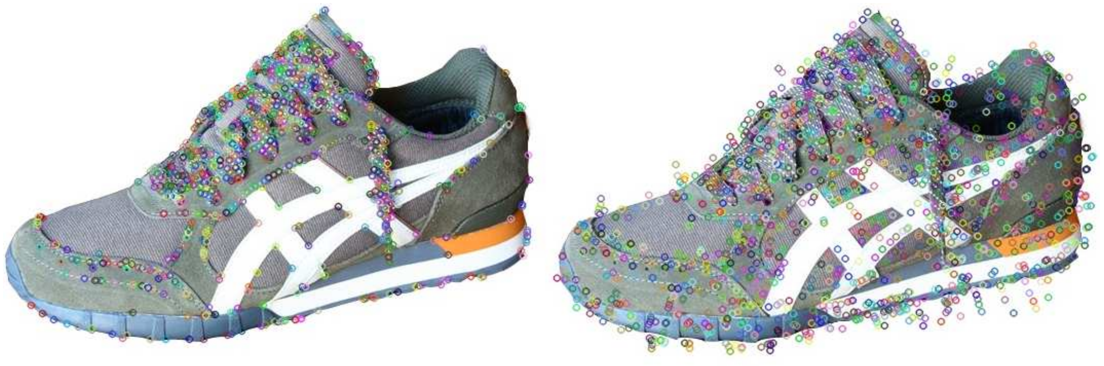


Figure 3.2: HARRIS (left) and SURF (right) key points detected in a query image

of local features. However, the local features are not directly analogous to the text words because they are feature points specific to the image. The vocabulary generation process, depicted in Figure 3.3, is needed to form the visual words. While generating the visual words, descriptors are extracted from the entire database images and they are clustered with k-means clustering. The centroids of these clusters become the visual words [64]. Thus, each local feature contributes to form the visual words through the clustering.

After getting the visual words and constructing the vocabulary, the images are represented as histograms that count the occurrence of the visual words in the specific image [25]. At the end, a large set of local features is mapped to fixed size histograms, as in the BoW approach. Histogram computation is explained in Figure 3.4. Assigning the local features to the visual words are done by finding the nearest neighbor. Each local feature increases the value of the histogram bin that corresponds to the assigned visual word. Figure 3.5 summarizes the two steps of $BoVW$ on a visual example [65].

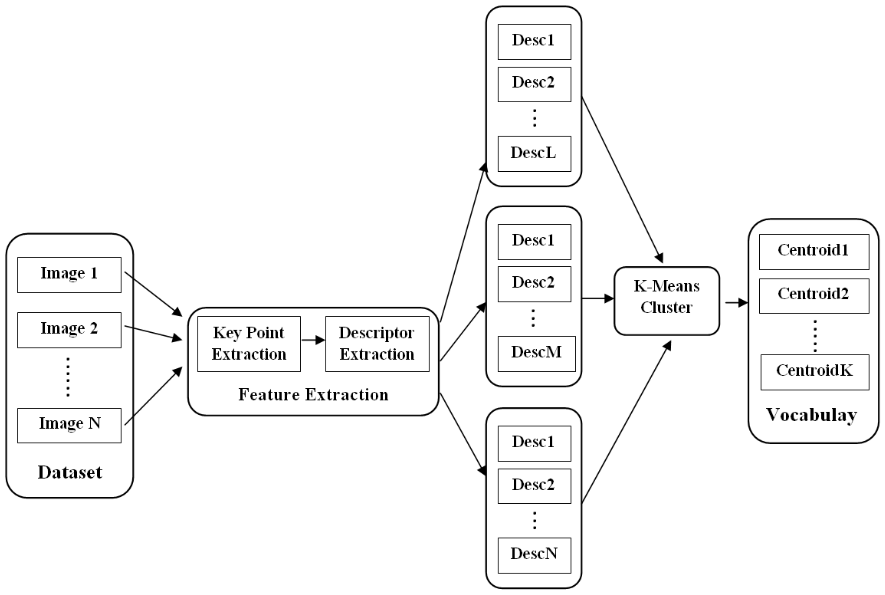


Figure 3.3: Generation of visual words that form the vocabulary

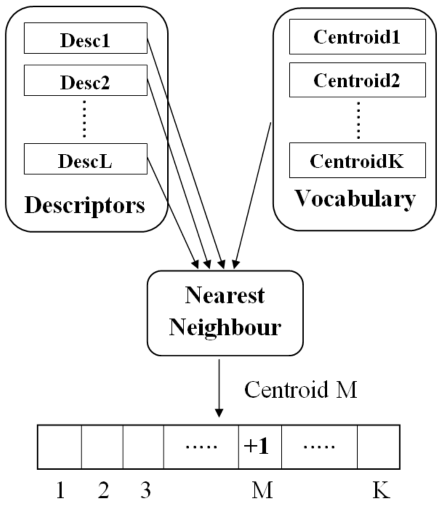


Figure 3.4: Histogram construction using centroids (visual words)

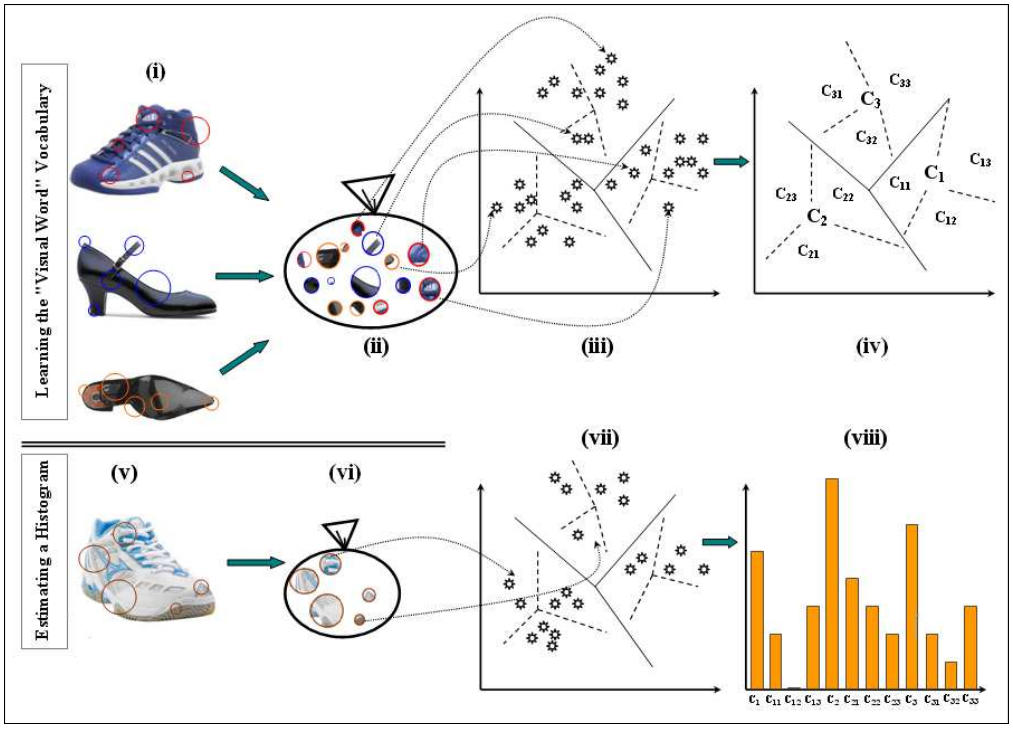

Learning the "Visual Word" Vocabulary
Estimating a Histogram
(i)
(ii)
(iii)
(iv)
(v)
(vi)
(vii)
(viii)
C1 C11 C12 C13 C2 C21 C22 C23 C3 C31 C32 C33

Figure 3.5: Illustration of the BoVW approach

3.3 Combining Multiple Features

The corners, edges and blobs describe different aspects of an image. While representing an image, using a single local feature focuses on a single aspect, because the feature uses either corners, edges or blobs. In order to increase the information collected from the query image, we use a multi key point approach. For each query, we use both *HARRIS* and *SURF* key points together. They use corners and blobs; this makes the representation include different aspects of the image. We extract both key points from the query image separately and combine them after creating the descriptors. The multi key point approach is described in Figure 3.6.

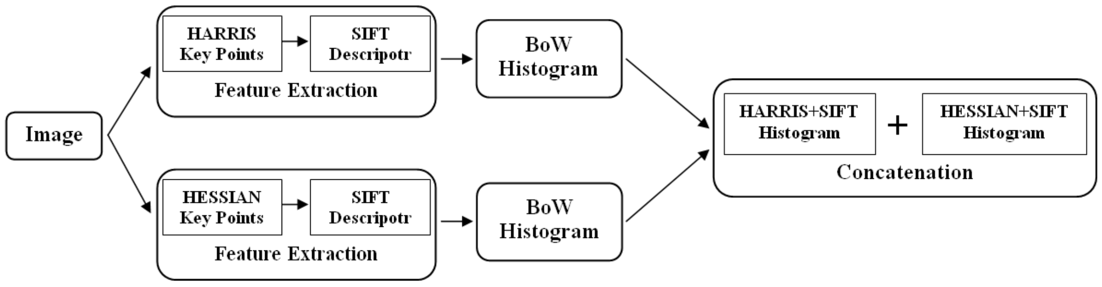


Figure 3.6: Combining HARRIS and SURF key points

3.4 Similarity Calculation

Another important point for matching is calculating distance (or similarity) between two images. After the quantization process, the images are represented as histograms constructed using features. While comparing histograms, it is better to calculate similarity rather than the distance between pairs of histograms [66]. There are various similarity metrics that can be used on histograms. The similarity metrics and their formulation are given in Table 3.1. In the table, h^q and h^d represent the histogram of the query and database images, respectively. During the calculation, q_i and d_i are the ith histogram bin of query and database histograms, respectively.

3.5 Important Parameters

Image search systems that use $BoVW$ approach have some parameters that must be set properly in order to have a high precision system.

Keypoint detector and descriptor: Key points and descriptors are the features used to represent the image. Their performance is directly related to the information collected from the image. If the features cannot represent the query image well, the image search system cannot give similar images as a

Table 3.1: Similarity metrics and their quality functions

Similarity Metric	Symbol	Quality Function
Normalized Correlation [55]	$NC(h^q, h^d)$	$\frac{\sum_i q_i d_i}{\sqrt{\sum_i q_i^2} \times \sqrt{\sum_i d_i^2}}$
Normalized Histogram Intersection [15]	$NHI(h^q, h^d)$	$\sum_i min\left(\frac{q_i}{\sum_i q_i}, \frac{d_i}{\sum_i d_i}\right)$
Histogram Intersection [9, 55]	$HI(h^q, h^d)$	$\frac{\sum_i min(q_i, d_i)}{min(\lvert h^q \rvert, \lvert h^d \rvert)}$
Dot Product [15]	$dot(h^q, h^d)$	$\sum_i q_i d_i$
Min-Max Ratio [67]	$MinMax(h^q, h^d)$	$\frac{\sum_i min(q_i, d_i)}{\sum_i max(q_i, d_i)}$

result.

Vocabulary size: While constructing the vocabulary, which is used to construct histograms of the images, the key points are clustered and the number of clusters (*vocabulary size*) is an important parameter. If it is high, then the key points cannot be separated well and the histogram becomes too broad. If it is low, then the related key points are separated from each other and the histogram becomes too specific. In either case, it affects the matching process in a bad way and the performance of the overall system decreases.

The similarity metric: The method used while comparing the query histogram with the database histograms directly affects the results because the images are labeled as *"similar"* according to the calculated similarity value. If the method is not good to measure the similarity between histograms, then the returned results cannot satisfy the user.

Chapter 4

Multi-Image Search

In our image search system, we take advantage of using multi-view image queries to increase the amount of information gathered from the query object. Multi image queries taken from different views and scales, depicted in Figure 4.1, help us understand and represent the query object better. The first image in Figure 4.1 gives information about the right side of the shoe object. When the second image is also used, we also gather information about the upper side of the shoe object. With every image shown in Figure 4.1, we increase the amount of information about the object.

The multi image query approach uses either the histograms or the results of the queries from the single image search system. In addition to these two approaches, we propose a new approach in which multi-view database images are used. We can sum up the query and database combinations in four categories:

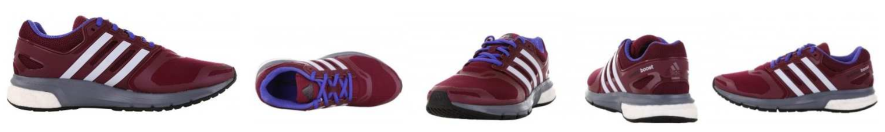


Figure 4.1: Multi-view image example

Single-view query and single-view database: In this case, both the query and database objects have a single image that represents a specific view of the object. The example images are depicted in Figure 4.2. The representation of the query and database images are limited to that view. For example, only the front or side view of a bag object can be represented as a query or database image. During the retrieval, the query image is compared to every database image to find best k matches.

Single-view query and multi-view database: This is the case where the query objects have single-view and database objects have multi-view images as shown in Figure 4.3. When the query image has limited amount of information, having multi view images in the database increases the performance of the search system. The reason is that using multi-view images in the database increases the probability of having similar images that can match to the query image. The matching is performed by comparing the query image to a set of database images that belong to the same object. A similarity score is then calculated for each image set of an object.

Multi-view query and single-view database: The query objects have multiple images, each representing a different view of the object.

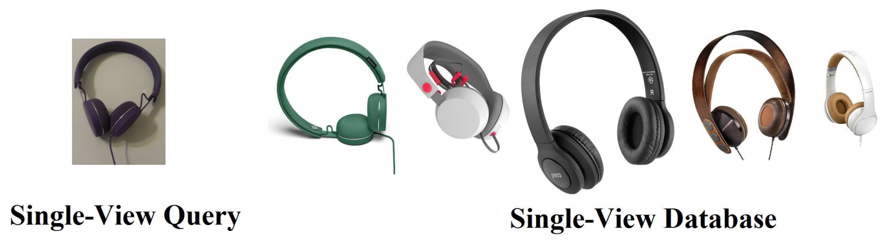


Figure 4.2: Example of single-view query and single-view database

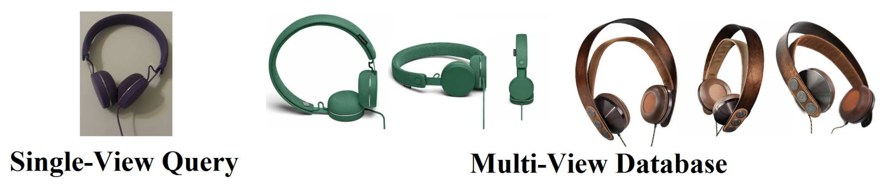


Figure 4.3: Example of single-view query and multi-view database

However, the database has a single image of each object. Example query and database images are presented in Figure 4.4. Either the queries are combined to form a single query, or the result list of each query is combined to obtain final results.

Multi-view query and multi-view database: In this case, as demonstrated in Figure 4.5, both the query and database objects have multiple images from different views. It is required to compare two sets of images of each object in the database, because both queries and each object in the database have multiple images. The amount of information gathered from the objects increases, but it also increases the matching time due to comparing not single but multi images of query and database.

4.1 Early Fusion

Early fusion, also referred to as fusion in feature space, is the approach where the histograms of the query images are combined into a single one. It is called early fusion because the queries are processed before they go into the image search process. The main idea here is that we increase our knowledge about the object by combining all histograms of the query images.

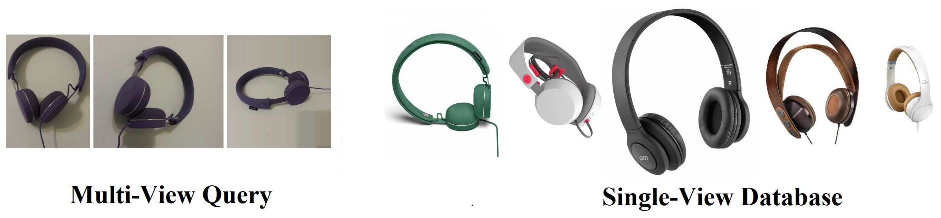


Figure 4.4: Example of multi-view query and single-view database

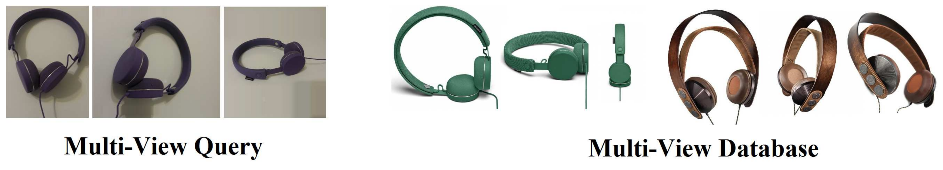


Figure 4.5: Example of multi-view query and multi-view database

Methods of early fusion are processing the query histograms and using the histogram bin values. A single histogram is created that contains information from all the other query histograms.

Early fusion requires a certain pre-processing like normalization in order to be sure that the features are on the same scale [5]. In our case, all the histograms are created by using the same feature extraction process; hence, pre-processing is not needed. Figure 4.6 depicts the early fusion approach to combine queries in feature space.

In our work, we implement and use three early fusion methods which are *Average Histogram*, *Maximum Histogram* and *Sum Histogram* methods [54, 55]. Table 4.1 summarizes the early fusion functions and their calculations. In the table, the histograms for M images are combined into h^c; h_i^j is the i^{th} bin of histogram h^j of image j.

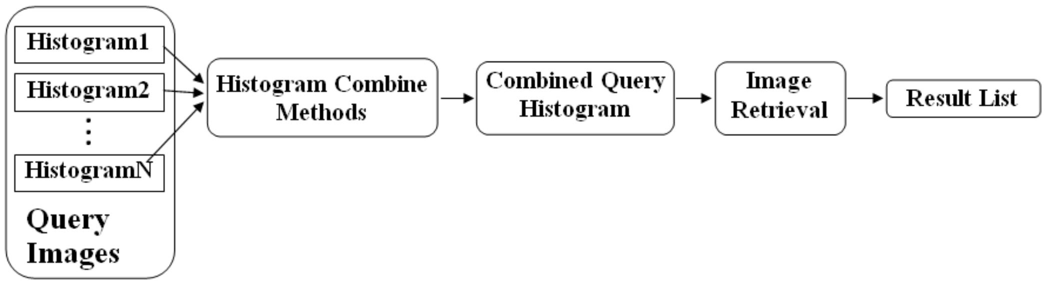


Figure 4.6: Early fusion approach for multi image query search

Table 4.1: Early fusion methods and the corresponding formulas

Method	Formula
Average Histogram	$h_i^c = \frac{\sum_{j=1}^{M} h_i^j}{M}$
Maximum Histogram	$h_i^c = max(h_i^1, \ldots, h_i^M)$
Sum Histogram	$h_i^c = \sum_{j=1}^{M} h_i^j$

4.2 Late Fusion

Late fusion, also referred to as decision level fusion, is the approach where the histograms of the query images are fed to the image search system and their result lists are combined into a single result list. Figure 4.7 depicts the late fusion approach to combine queries at decision (*result list*) level.

It is called late fusion because the queries are processed and then their results are combined. The main idea here is that by combining the result lists of the query histograms, we benefit from the information that each query image transfers to its result list. Late fusion approaches process the result lists, not the individual histograms, and they merge them into a single result list, which has all the information gathered from the query images separately. In our work, we implement and use the following late fusion methods [54, 68].

- *Max Similarity (MAX SIM):* Each database image is compared with the query images and the similarity is taken as the maximum of the similarities.
- *Weighted Similarity:* Each database image is ranked according to a weighted similarity to the query images. The weight is calculated as the ratio between the current and total similarity values.
- *Count:* For multiple query images, multiple result lists are obtained. Then, for each image, a counter is incremented if it is in a list. Finally, the counter value is used to rank the database images. The higher value means the

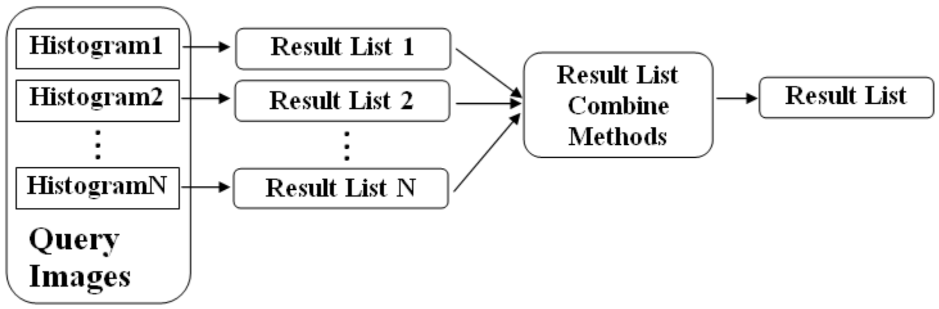


Figure 4.7: Late fusion approach for multi image query search

higher rank.

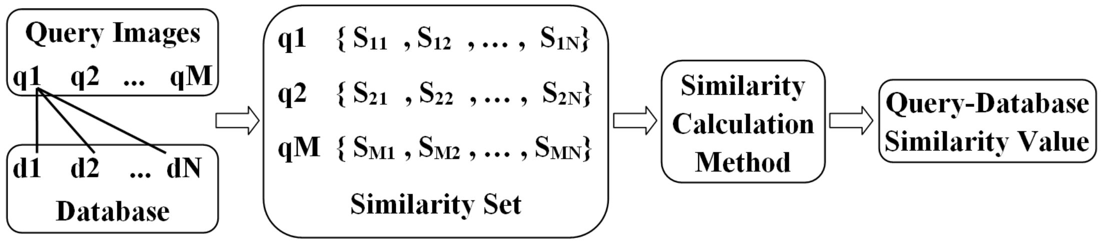
- *Highest Rank:* For multiple query images, multiple result lists are obtained and the highest rank is taken for each database image.
- *Rank Sum:* For multiple query images, multiple result lists are obtained and the ranking of each image in every list is summed and the resulting values are used to rank the database images.

4.3 Matching Image Sets

In addition to the query objects, representing the database images is also important for the performance of the image search system. Increasing the amount of information collected from the database images enables us to perform better matching between the query and database images. We think that having multi-view images in the database may increase the overall performance of the image search system.

This new approach brings some modification to our previous image search system in terms of the objects in the database. The difference is that the objects are not stored as a single image anymore; hence, the matching methods must be modified. Previously, we had a multi image query set that has two or more images and we were comparing the set with every image stored in the database. Now, we need to compare two image sets both having multiple images.

In order to measure the similarity between the query and database image sets, we need to calculate a single similarity value. Therefore, we first need to calculate the similarity values between the query and database images, then using these values we calculate a single similarity value that represents the similarity between the query and the database. The similarity calculation is demonstrated in Figure 4.8.



Query Images
q1 q2 ... qM
d1 d2 ... dN
Database
q1 { S11 , S12 , ... , S1N}
q2 { S21 , S22 , ... , S2N}
qM { SM1 , SM2 , ... , SMN}
Similarity Set
Similarity Calculation Method
Query-Database Similarity Value

Figure 4.8: Similarity calculation between image sets

In this approach, various similarity calculation methods that transform the set of similarity values into a single similarity value can be used. We propose five different methods for this purpose.

4.3.1 Average Similarity

This method takes the average of the similarity values to assign a single similarity value between the query and database image sets. Taking the average decreases the effect of the query that has a low similarity. The calculation is trivial and shown in Equation 4.1. In the equation, S_{ij} represents the similarity value between the query image i and database image j.

$$Similarity = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} S_{ij}}{M \times N} \tag{4.1}$$

4.3.2 Weighted Average Similarity

In this approach, a weight is assigned to each similarity value in the similarity set. We then take a weighted average of the similarity values where a query that has a high similarity has a high effect on the similarity calculation. The calculation is shown in Equation 4.2. In the equation, S_{ij} represents the similarity value between the query image i and database image j. Similarly, W_{ij} is the weight assigned to the relation between the query image i and the database image j. The final similarity function that shows the similarity between the image sets is $single\_sim$.

$$\begin{aligned} W_{ij} &= \frac{S_{ij}}{\sum_{i=1}^{M} \sum_{j=1}^{N} S_{ij}} \\ Similarity &= \sum_{i=1}^{M} \sum_{j=1}^{N} S_{ij} \times W_{ij} \end{aligned} \tag{4.2}$$

4.3.3 Maximum Similarity

The main idea of this method is taking advantage of the query image that has the maximum value. When the similarity set is formed by calculating the pairwise similarity values between the query and database images, it may have misleading information. The query images that are not taken from a good angle or have blurry parts have lower similarity value. The reason is that, these images are not represented well, because the local feature descriptors work poorly on the images. When considering all the values in the set, the calculated overall similarity value also has the negative effect of badly represented query images. In order to get rid of these values, the maximum value is selected from the similarity set. After the similarity set is calculated, they are sorted in descending order. The maximum value is then selected as the query-database similarity value. The calculation is simple and shown in Equation 4.3, where S_{ij} represents the similarity value between the query image i and database image j.

$$Similarity = max(S_{ij}) \tag{4.3}$$

4.3.4 Average Maximum Similarity

This method takes advantage of using maximum value (*maximum similarity method*) and average value (*average similarity method*). First of all, the maximum similarity value is taken to increase the effect of the query images that have higher similarity. Then, the average of the maximum similarity values is taken to reduce the effect of dissimilar query images. The calculation is shown in Equation 4.4. In the equation, S_{ij} represents the similarity value between the query image i and database image j.

$$Similarity = \frac{\sum_{i=1}^{M} \max(S_{i1}, \ldots, S_{iN})}{M} \tag{4.4}$$

4.3.5 Weighted Average Maximum Similarity

This method is similar to *average maximum similarity* method. The only difference is taking weighted average instead of normal average. With this approach, the effect of similar query images is increased. The calculation is shown in Equation 4.5. In the equation, S_{ij} represents the similarity value between the query image i and database image j. Similarly, W_{ij} is the weight assigned to the relation between the query image i and the database image j.

$$\begin{aligned} S_i &= \max(S_{i1}, \ldots, S_{iN}) \\ W_i &= \frac{S_i}{\sum_{i=1}^{M} S_i} \\ Similarity &= \sum_{i=1}^{M} W_i \times S_i \end{aligned} \tag{4.5}$$

4.4 Query Processing

The image search system we developed is a mobile application that allows users to perform multi image query search. One of the motivations behind our work is that it is easy to provide multiple queries to the image search system by using the built-in camera of the mobile device. The user interface of the application is simple and user friendly. In order to search for an object, users need to take a photo of the desired object. If the user does not like the taken photo due to illumination or blurring problems, the taken photo can be discarded and a new photo can be taken. If the quality of the photo is acceptable to the user, there are two options at that point. The user goes with that image or more photos of the object can be taken to provide multiple queries. If the user wants to take more photos, the camera is automatically opened and the user is allowed to take more photos.

One of the drawbacks of multi image query search systems is the execution time. Increasing the number of queries also increases the execution time. In order to reduce the execution time and increase user satisfaction, the application is designed to process the queries in a parallel way. The parallelization process shows some differences according to the used multi image query approach. The main workflow of the mobile application, when early or late fusion approach is used, is depicted in Figure 4.9.

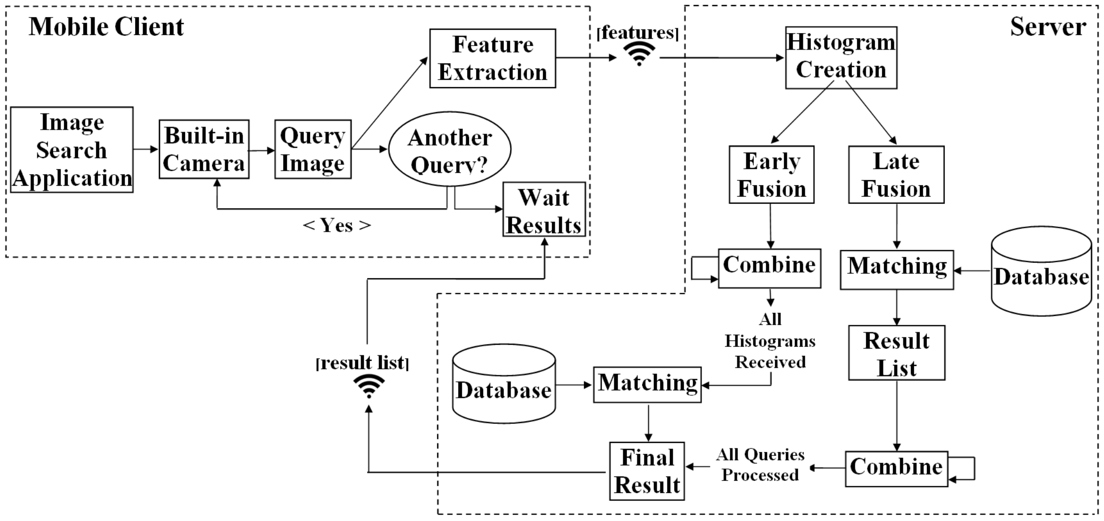


Figure 4.9: Workflow of our image search system using early and late fusion methods

Independent from the choice of the fusion approach (early or late), when a photo is taken and accepted by the user, key points are extracted and corresponding descriptors are calculated in the mobile device. The extracted descriptors are then sent to the server via the Internet connection. When the descriptors are received by the server, their histograms are constructed using the visual vocabularies. Similarly, histogram construction is performed independently from the fusion approach. While the user is taking another photo of the object, the first one is already processed, its features are extracted, and the histogram is constructed accordingly. This is the first part of the parallelism applied by our mobile application. The rest of the process shows some differences according to the choice of the fusion approach.

4.4.1 Query Processing for Early Fusion

When the early fusion approach is applied to combine the queries, all the query histograms are needed on the server side. As the query photos are taken, their features are sent to the server and histograms are constructed. When a new histogram arrives at the server, it is immediately combined with the one that belongs to the previous query image. For example, when the first query photo is taken, it is sent to the server and its histogram is constructed. Then, when the second photo is taken, it is also sent to the server and immediately combined with the first query histogram. From that point, only the combined histogram of the first and second query photos is kept on the server. When a third query photo is taken, it is sent to the server and its histogram is combined with the histogram, which is the combination of the first and second query photos. This means that the multi image query approach only requires an additional time to combine the histogram of the last query photo. The histograms of previous query photos are already combined together when the last query photo arrives at the server. The rest of the search process is the same with the single query approach. The combined histogram is compared with the ones stored in the database. Then the result list is sent to the mobile client and shown to the user.

4.4.2 Query Processing for Late Fusion

The late fusion requires to combine the result lists of the query histograms. In order not to lose extra time, when a query photo is taken, its features are sent to the server and the histogram is constructed. In the late fusion approach, the server side continues to process the histogram and creates the result list as a result of the matching. When another query photo is taken, it goes through the same processes and the newly formed result list is combined with the previous one. That means, when a query photo is taken, its result list is calculated and combined with the previous result list, which is also a combination of previous result lists. In total, the multi query approach requires only an additional time to combine the result list of the last query photo. The result lists of previous query photos are already combined together when the last query is processed. The final combined result list is then sent to the user as the result of the image search.

4.4.3 Query Processing for Multi-View Database

The workflow of the search system is different when the multi view database images are used. The mobile client side is the same as the previous cases. However, the server side differs from the others. The reason is that, in early and late fusion cases, each database image is processed separately. When the multi-view database images are used, every object in the database has multiple images, each representing a different view of the object. The process is depicted in Figure 4.10

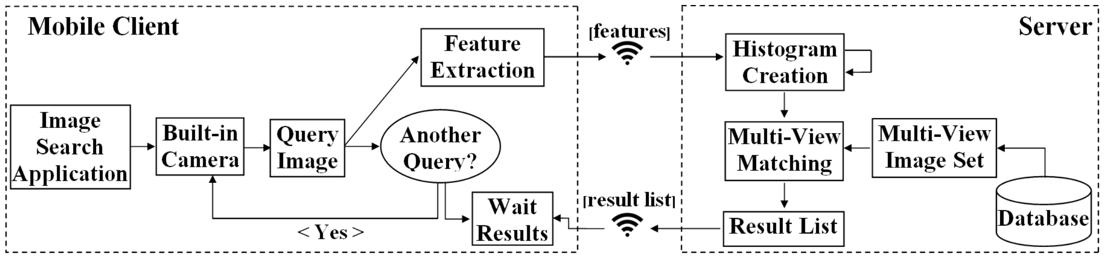


Figure 4.10: Workflow of our image search system using early and late fusion methods

Chapter 5

Performance Evaluation

5.1 Evaluation Metric

We evaluate the proposed image search system in terms of *average precision* value [69] which is calculated as shown in Equation 5.1. In the equation, k represents the rank in the sequence and N is the length of the result list.

$$
\begin{aligned}
P(k) &= \frac{relevant\ images \cap first\, k\, images}{k} \\
rel(k) &= \begin{cases} 1, & \text{if image k is relevant} \\ 0, & \text{otherwise} \end{cases} \\
AveP &= \frac{\sum_{k=1}^{N}(P(k) \times rel(k))}{N}
\end{aligned}
\tag{5.1}
$$

5.2 Datasets

During the evaluation of the proposed image search system, we use two image datasets.

5.2.1 Caltech-256

The first database used is a subset of *Caltech-256* database, which is used in [2] and has the following properties:

- It has 20 categories and 844 objects;
- Objects are positioned at the image center and have a clean background;
- It has total 60 query images from six categories as query images;
- Query images contain background clutter;
- Images are downloaded from Google Images [70].

Some sample images are demonstrated in Figure 5.1.

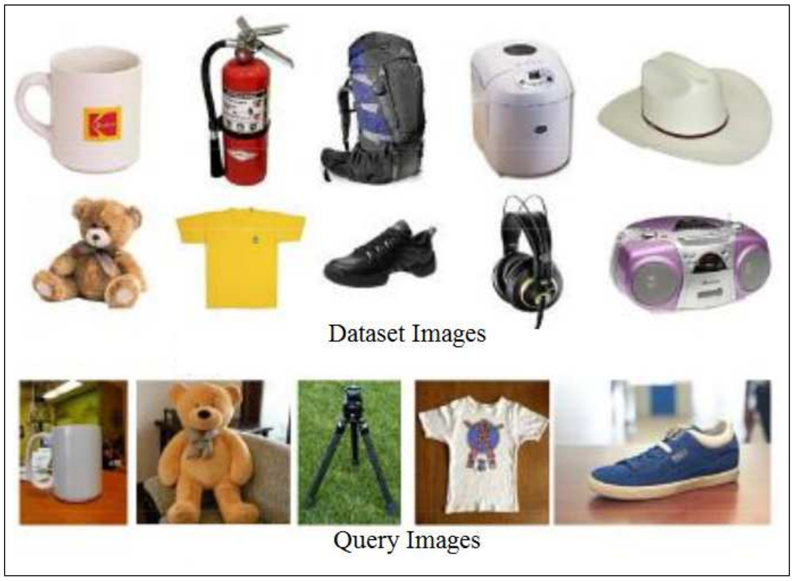


Figure 5.1: Example query and database images from Caltech-256 database

The reason to choose this database is that we want to compare our results with the work of Shen et al. [2], and thus we need to use same image database and query images. They provide performance results obtained using the $BoVW$ and single query image search approaches.

5.2.2 Multi-View Object Images Dataset (MVOD)

Caltech-256 database contains images that are downloaded from the Internet and the images belonging to the same category are different objects. For example, the bags category has images of different bag objects. Our image search system is using a multi query approach. We think that having a database that has multi view objects may increase the performance of the search system. Therefore, we collected images from the Internet, but for each object of each category, we have at least two images which are different views of the same object. This collected database is more suitable for the multi-view query approach, because the query images which are taken from different views can match to the multi view database images easier. Some sample images are demonstrated in Figure 5.2.

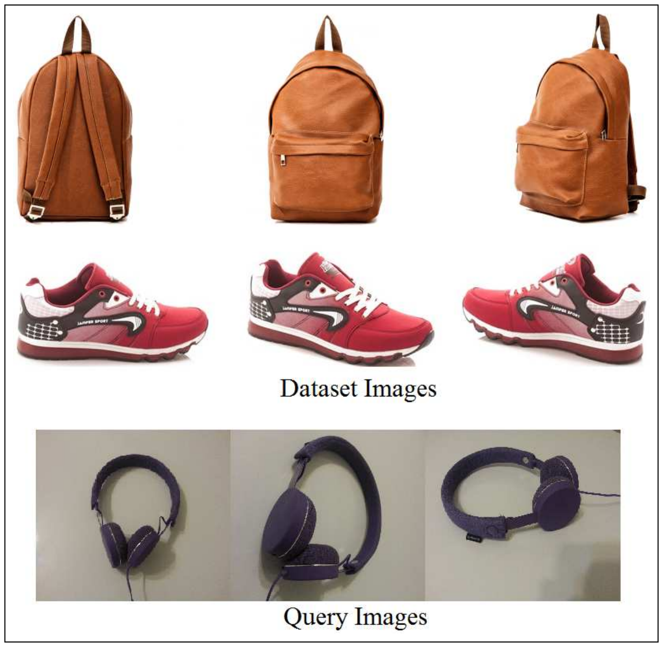


Figure 5.2: Example query and database images from MVOD database

MVOD database has the following properties:

- It has 6 categories and 1664 images;
- Objects are positioned at the image center and have a clean background;
- Objects have multiple view images;
- It has total 30 query images from four categories as query images;
- Queries are the images taken by a mobile phone;
- Images are downloaded from the Internet.

5.3 Experimental Results

We provide the performance results of our image search system using Caltech256 and MVOD datasets. We used the OpenCV library [71] to extract the local features. The vocabulary size is 3000. Some parts of the results gathered by using *Caltech-256* image dataset are compared to the results provided in [2]. We also analyze the effect of background cluttering on query images. The performance of the multi image query approach relative to single image query approach is also examined. The experimental results obtained using *MVOD* database show the performance impact of multi-view database and queries. Furthermore, the effect of using multi image queries in terms of matching time is evaluated.

5.3.1 Test Environment

In order to evaluate performance of our image search system, we have prepared four different test environments by using two different databases and two different types of queries.

- *Caltech-256* database by using background cluttered queries,
- *Caltech-256* database by using queries with clear background,
- *MVOD* database by using background cluttered queries,
- *MVOD* database by using queries with clear background.

5.3.2 Results on the Caltech-256 Dataset

As we mentioned before, *Caltech-256* database is a single-view database where each database object has a single view image. For the queries, we use both single-view and multi-view images during the experiments. Therefore, the following results include the combination of single-view query and database, and multi-view query and single-view database.

We use both background cluttered queries and queries with clean background to make our results comparable to those of Shen et al. [2], as they also use the same types of queries in their evaluation. The results can be seen in Figure 5.3.

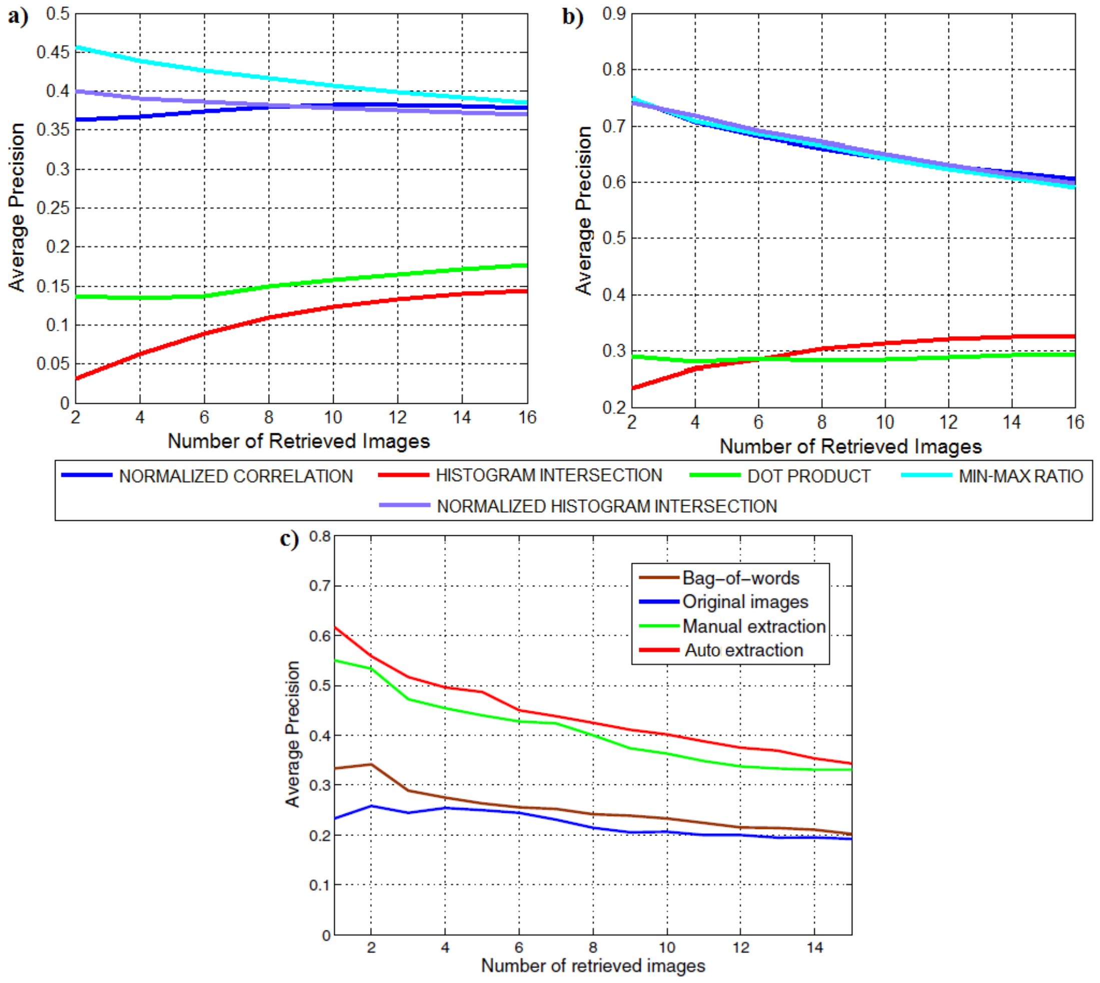


Figure 5.3: The average precision of queries using our approach for background cluttered (a) and clean background (b), and the average precision results from [2] (c).

First of all, we need to compare Figure 5.3 *(a)* and bag-of-visual-words ($BoVW$) approach (*brown*) of Figure 5.3 *(c)*. It is reasonable to compare them because both works use the bag-of-visual-words approach and background cluttered images. When the results are compared, it is seen that our best result, which is obtained by using Min-Max Ratio similarity function, has 0.15 higher average precision value than the result in Figure 5.3 *(c)*. Considering that the database and query images are the same and $BoVW$ approach is used, we conclude that there are two reasons that make our system better in terms of precision:

- We better adjust the parameters that are important for the overall performance of the image search system. We perform a detailed analysis on the image database to find out the best choices for the parameters explained in Section 3.5. The selected parameter set is as follows:
  - Key point detector: *HARRIS* and *SURF*;
  - Key point descriptor: *SIFT*;
  - Vocabulary size: 3000;
  - Similarity function: *Max-Min Ratio*.
- We use a combined feature extraction approach. While creating the histograms of the images, we use both of the key point extraction methods (*HARRIS* and *SURF*), which are selected according to the detailed analysis of the image database. Combining the results of these two key point extractors increases the information gathered from the query images and leads to better performance.

The visual results of the background cluttered query is depicted in Figure 5.4. The results are in accordance with the results shown in Figure 5.3 *(a)*. The *Min-Max Ratio* function gives better results than *Normalized Histogram Intersection* function. The result list in Figure 5.4 *(a)* has more similar images and better ranking than the result list in Figure 5.4 *(c)*.

The other comparison is made between Figures 5.3 *(b)* and *(c)*. This time the focus is on *Manual Extraction (green)* and *Auto Extraction (red)* approaches in which [2]

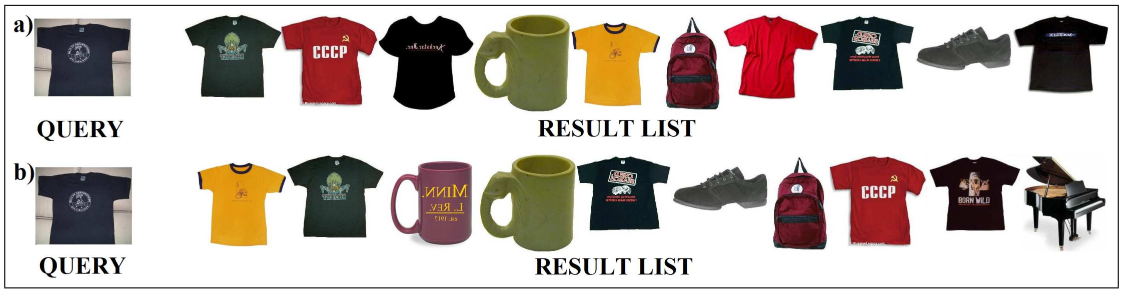


Figure 5.4: The results of the background cluttered query by using Max-Min Ratio (a) and Normalized Histogram Intersection (b) similarity functions

uses the queries with a clean background. In the experiments whose results are shown in Figure 5.3 *(b)*, we also use the queries with a clean background, in order to have compatible and comparable results. The figures show that the best result we get has 0.2 higher precision value. We can say that our image search system performs better also when queries with clean background are used. These results are important because they show that the base (single query approach) of our image search system works well due to the aforementioned reasons.

The visual results of the query with clean background are shown in Figure 5.5. The results are parallel to the results shown in Figure 5.3 *(b)*. The results of *Min-Max Ratio* and *Normalized Histogram Intersection* functions are very close to each other. The result list in Figure 5.5 *(a)* has one more similar object than the result list in *(c)*, and the ranking of the similar objects is same in both result lists.

After the experiments with single query approach, we also conduct some experiments with the multi query approach and combined methods. Similarly, we use queries both with and without background clutters and apply early and late fusion methods in our image search system. The queries used in the experiment are multi image queries. Each query type has multiple images representing different views and scales. Then we compare the results (shown in Figure 5.6) with the results obtained with the single image query approach. The dashed green line shows the best result of the single query approach. The results in Figure 5.6 *(a)* are obtained by using queries with background clutter and the Figure 5.6 *(b)* by using queries without background clutter. The results show that the multi query

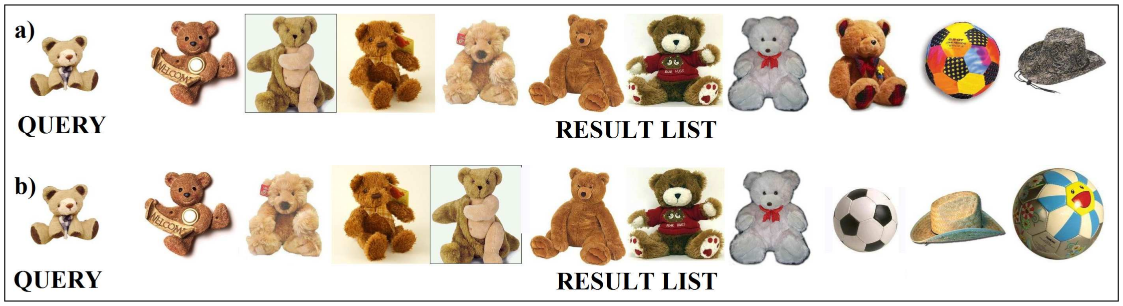


Figure 5.5: The results of the background cluttered query by using Max-Min Ratio (a) and Normalized Histogram Intersection (b) similarity functions

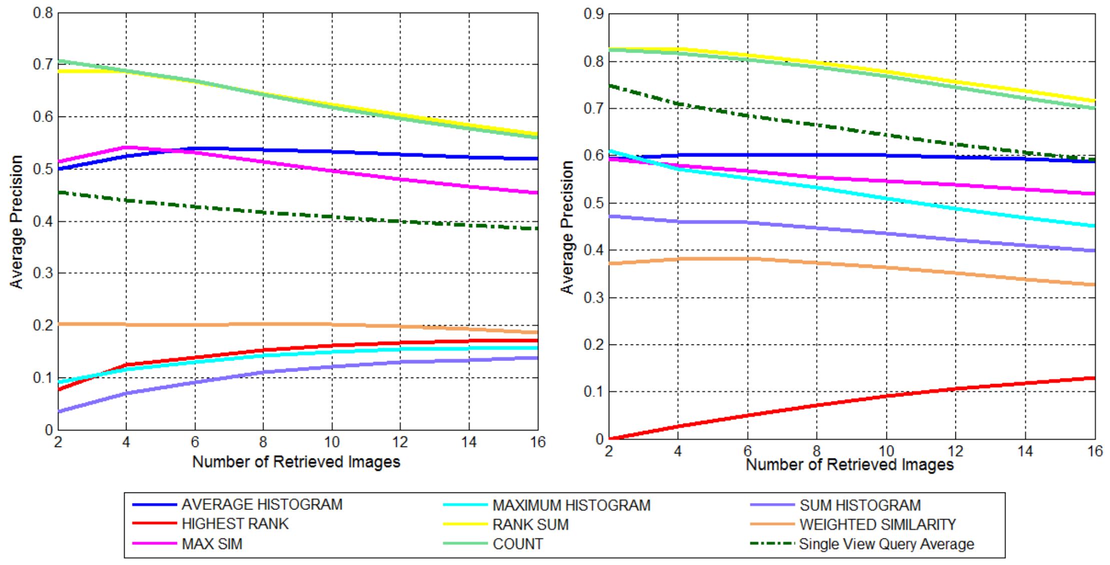


Figure 5.6: Results with background cluttered queries (a) and queries with clean background (b) using early and late fusion methods

approach increases the precision by 0.25 when queries with background clutter are used, and by 0.1 when queries without background clutter are used. These results prove that multi query approach provides more information about the query object and increases the performance of the system in terms of precision.

The amount of increase in precision differs according to the query type. The increase is higher when queries have background clutter. When there is background clutter in the query, it misleads the feature extraction method because the features come from the background are also considered as the features of the query. When we use multiple photos, the information gain from the object becomes higher than the misleading information which comes from the background. Therefore, we represent the query object better.

The performance of the combined methods differs according to the type of the query used. However, *Rank Sum* and *Count* methods, which belong to the late fusion category, increase the performance independent from the query type used. Therefore, these methods can be selected for the multi query image search system. The visual example, shown in Figure 5.7, illustrates the impact of using multi image query and late fusion methods (*Rank Sum* and *Count*). The result lists

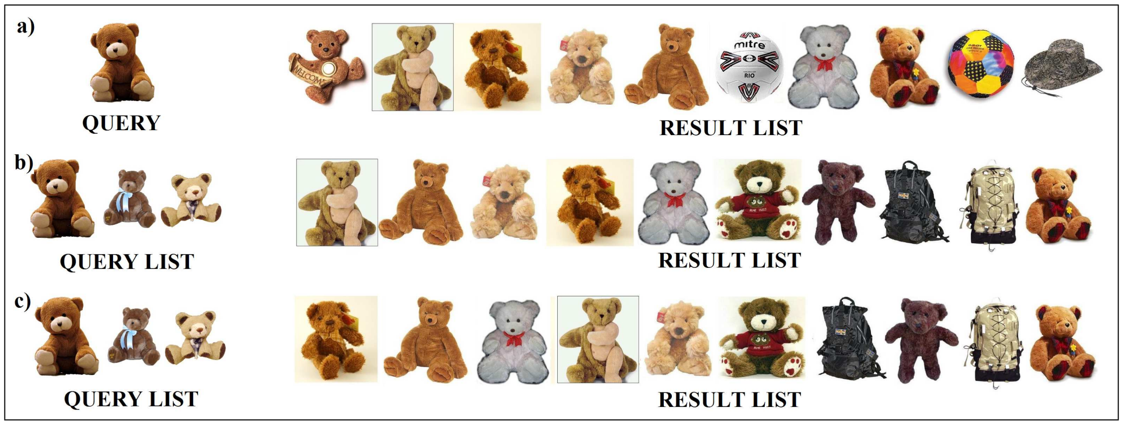


Figure 5.7: Results of single image query (a) and multi image query combined by using Rank Sum (b) and Count (c) late fusion methods

obtained using late fusion methods are shown in Figure 5.7 *(b)* and *(c)* have better result lists than single query image shown in *(a)* in terms of, ranking and the number of similar images. Figure 5.7 also proves that using multi image queries together with late fusion methods *Rank Sum* and *Count* increases the performance of the image search system.

Figure 5.8 *(a)* and *(b)* present the result lists of two different early fusion methods which are *Average Histogram* and *Maximum Histogram*, respectively. Although the result lists in Figure 5.8 *(a)* and *(b)* have same number of similar objects, the ranking in *(a)* is better than *(b)*, which proves that the *Maximum Histogram* method increases the performance of the image search system.

Up to this point, we use the queries from *Caltech-256* database. However, our

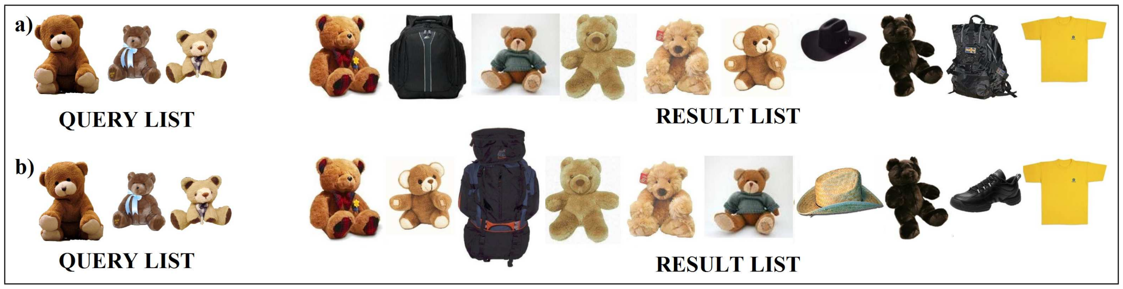


Figure 5.8: Results of early fusion methods: Average Histogram (a) and Weighted Average Histogram (b)

image search system takes the queries from the camera of the mobile device. Testing the system with query images taken by the mobile device gives us more reliable results in terms of the performance of the mobile image search system. We take photos of some images with the camera on a mobile device; and the results are shown in Figure 5.9.

As we can see, the highest precision value is achieved with this experiment. The reason is that the queries used in previous experiments are the different images of the same category. However, in this experiment we took multiple photos of the same object from different views. The information collected from multiple images becomes more coherent and we achieve the highest precision value which is 0.90 on the average.

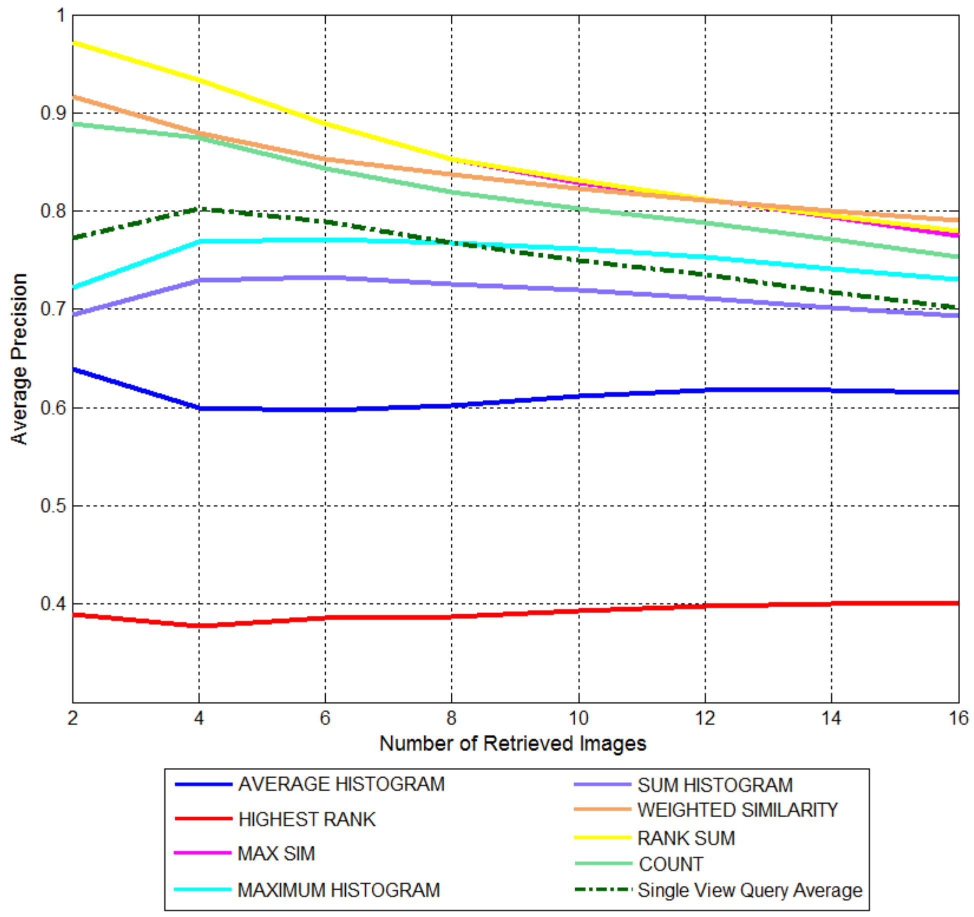


Figure 5.9: Results of using query photos taken by mobile phones and combine methods

The multi-view query photos taken by mobile phone and the result list are shown in Figure 5.10. The results show that the highest number of similar objects and best ranking is achieved compared to Figure 5.7 and Figure 5.8. There are nine similar objects out of ten, and only dissimilar object is at the end of the list. We can conclude that our image search system is well designed for the mobile users and the environment.

5.3.3 Results on the MVOD Dataset

In this section, we present the performance results by using multi image queries together with multi-view database images. We use both background cluttered queries and queries with clean background as the experiments performed using *Caltech256* database. In the experiments, we use 15 images for both queries with and without background clutter. The results can be seen in Figure 5.11.

First of all, we need to compare Figure 5.11 *(a)* and Figure 5.3 *(a)*. In both experiments, the query images are single-viewed and have background clutter. When we compare the best average precision values, which are obtained using *Min-Max Ratio* similarity function in both experiments, we observe that the average precision using *MVOD* database is 0.1 higher. The only difference between the experiments is the used database. Using *MVOD* database, which has multi-view query images, increases the overall performance of the image search system.

The visual results of the background cluttered query is depicted in Figure 5.12. The results are in accordance with the results shown in Figure 5.11 *(a)*. The *Min-Max Ratio* function gives better results than *Normalized Histogram Intersection* function.

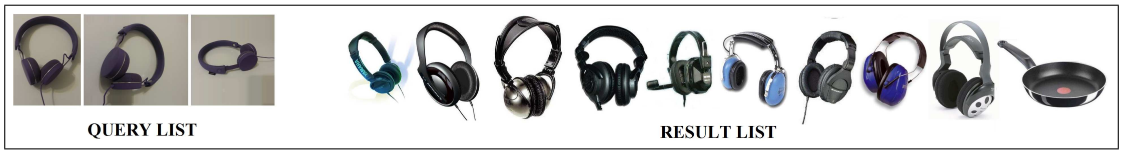


Figure 5.10: Visual example of the combined result using multi query photos taken by mobile phone

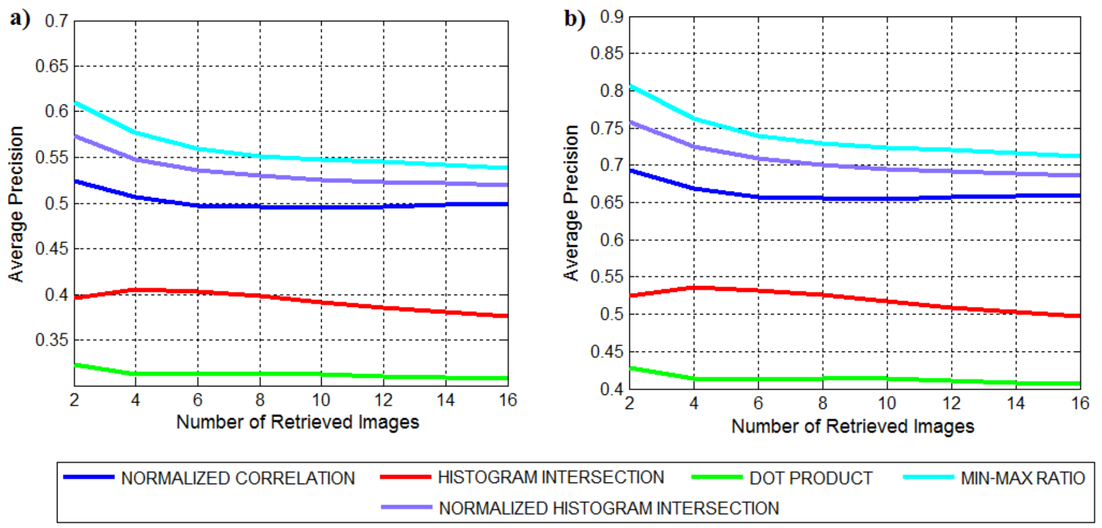


Figure 5.11: The average precision of our image search system using single-view queries which are background cluttered (a) and having clean background (b).

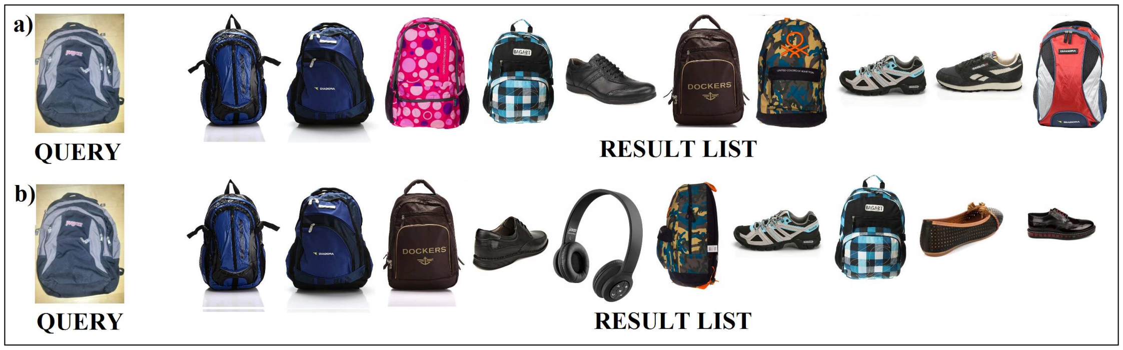


Figure 5.12: The results of the background cluttered query by using Max-Min Ratio (a) and Normalized Histogram Intersection (b) similarity functions

We also compare the results depicted in Figure 5.11 *(b)* and Figure 5.3 *(b)*, where the single-view query images with clean background are used. The best results obtained using *Min-Max Ratio* similarity function show that using *MVOD* database increases the average precision by 0.1 compared to the results obtained using *Caltech256* which is a single-viewed database. We can conclude that using the proposed multi-view database increases the average precision of the image search system using queries with and without background clutter.

The visual results of the query with clean background are shown in Figure 5.13.

The results are parallel to the results shown in Figure 5.11 *(b)* where *Min-Max Ratio* function has better retrieval performance (*average precision*) then *Normalized Histogram Intersection* function. Same results are shown in Figure 5.13 where *(a)* has more similar objects and better ranking.

After the experiments with single query approach, we also conduct some experiments using multi-view query images and fusion methods. Similarly, we use queries both with and without background clutters and apply early and late fusion methods in our image search system. The results are depicted in Figure 5.14. The experiment is similar to one performed using *Caltech256* database. The results in Figure 5.14 *(a)* are obtained by using queries with background clutter and the Figure 5.14 *(b)* by using queries without background clutter. The dashed green line in *(a)* and *(b)* shows the best results of the single query approach.

Figure 5.14 *(a)* and *(b)* show that the multi query approach increases the precision by 0.25 when queries with background clutter are used, and by 0.1 when queries without background clutter are used. These results prove that multi query approach provides more information about the query object and increases the performance of the system in terms of precision.

We also compare the results depicted in Figure 5.14 and 5.6 in order to observe the effect of using multi-view database (*MVOD*) in our image search system.

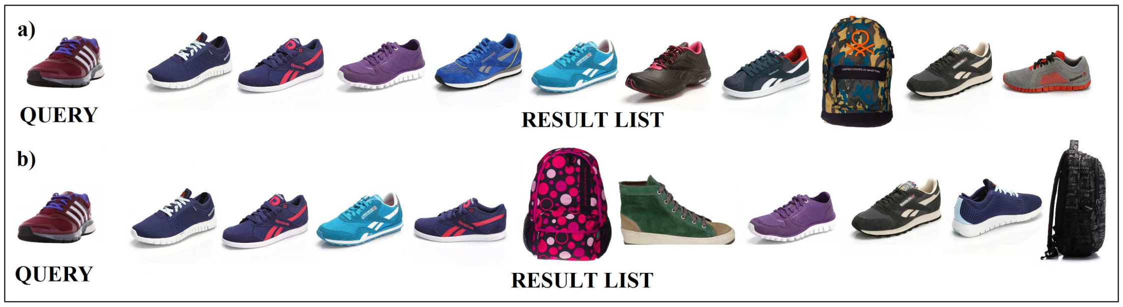


Figure 5.13: The results of the background cluttered query by using Max-Min Ratio (a) and Normalized Histogram Intersection (b) similarity functions

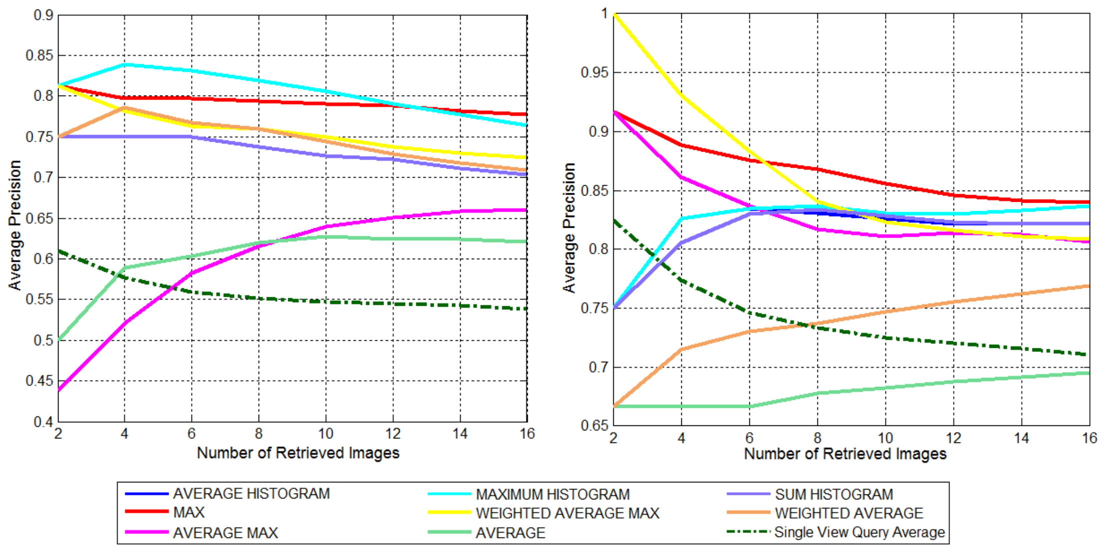


Figure 5.14: Results with background cluttered queries (a) and queries with clean background (b) using early and late fusion methods

Figure 5.6 shows that the fusion methods increases the average precision up to 0.7 and 0.80, when using queries with and without background cluttered queries, respectively. When *MVOD* database is used, the average precision is increased up to 0.82 with background cluttered query images, and 0.92 with queries having clear background. Using multi-view query images and *MVOD* database in our image search system further increases the retrieval performance.

The early fusion methods used in the experiments are same with the ones used in experiments with *Caltech256* database. However, the late fusion methods are different because the *MVOD* database has multi-view images which requires late fusion methods to obtain result lists from query and database image sets. Therefore, different set of late fusion methods, which are explained in Section 4.3, are used.

The performance of the combined methods differs according to the type of the query used. However, *Max* and *Weighted Average Max* methods, which belong to the late fusion category, increase the performance independent from the query type used. Therefore, these methods can be selected for the multi query image search system. The visual example, shown in Figure 5.15, illustrates the impact of using multi image query and late fusion methods (*Max* and *Weighted Average Max*). The result lists obtained using late fusion methods are shown in Figure 5.15 *(b)* and *(c)* have better result lists than single query image shown in *(a)* in terms of, ranking and the number of similar images.

Figure 5.16 *(a)* and *(b)* present the result lists of two different early fusion methods which are *Maximum Histogram* and *Average Histogram*, respectively. These early fusion methods are the ones which gives the highest average precision value.

In conclusion, the results prove that using multi-view images improve the performance of the image search system. We give some visual examples with different multi-view queries in Figures 5.17, 5.18 and 5.19, where *(a)* shows the results of single-view query, *(b)* and *(c)* show the results of *Max* and *Weighted Average Max* methods, respectively. These results are obtained using multi-view database images; the methods are used to calculate the similarity between the query and the database images. All the figures prove that the performance of the image search system increases when multi-view database images are used. In all the figures, when the multi-view query is used together with the multi-view database images, the number of similar objects and their ranking are improved.

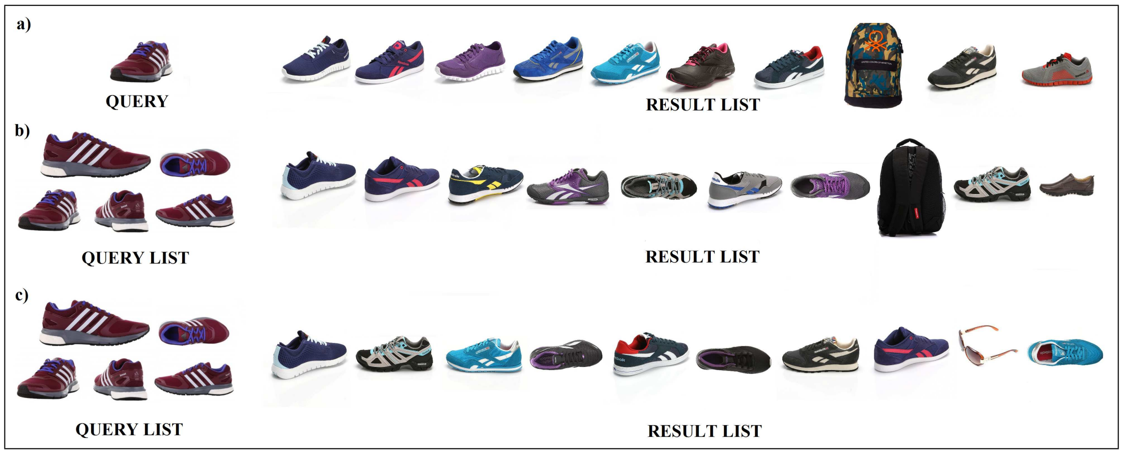


Figure 5.15: Results of single image query (a) and multi image query combined by using Max (b) and Weighted Average Max (c) late fusion methods

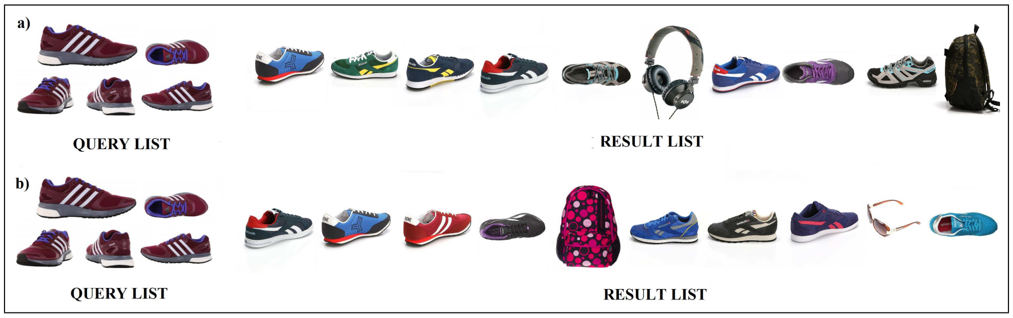


Figure 5.16: Results of early fusion methods: Maximum Histogram (a) and Average Histogram (b)

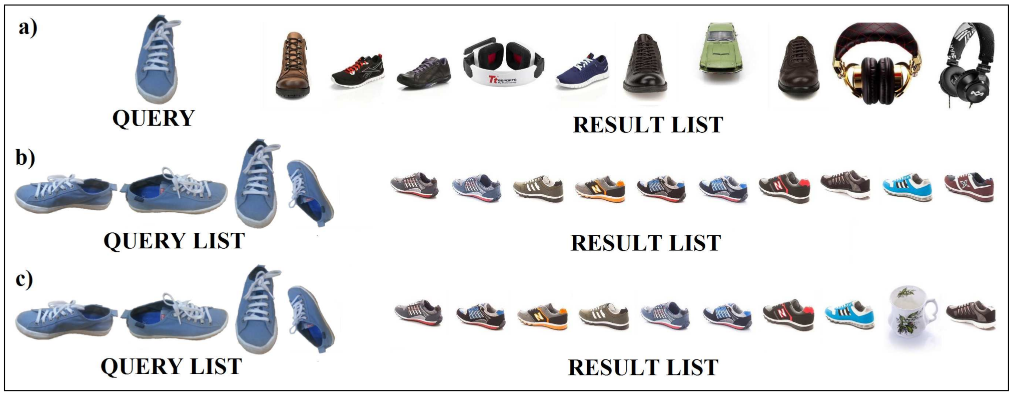


Figure 5.17: The results of single-view query object shoe (a), and Max (b), and Weighted Average Max methods (c), using multi-view database images

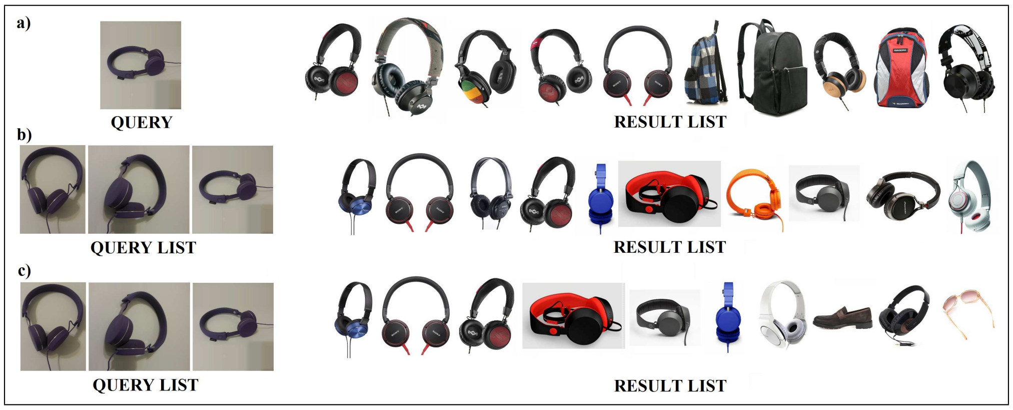


Figure 5.18: The results of single-view query object headphone (a), and Max (b), and Weighted Average Max methods (c), using multi-view database images

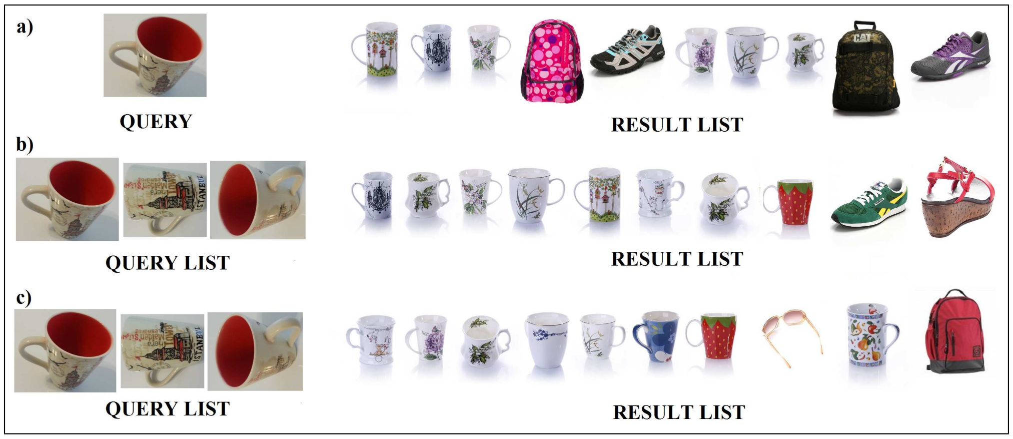


Figure 5.19: The results of single-view query object mug (a), and Max (b), and Weighted Average Max methods (c), using multi-view database images

5.3.4 Matching Time Comparison

In order to observe the effects of multi view image query at the time spent during matching, we calculate and compare the matching time. The time measurement is performed on an *Intel Xeon E5 3.30GHZ* machine by using *java.lang.management* package and calculating *CPU time*. In addition to the multi image query approach, we also analyze the time effect of using different similarity functions. We analyze only the matching time because the multi image query approach mainly increases the time spent on matching. We also detect the similarity function which is suitable for multi image query approach considering the spent matching time. The matching time observed when using different type of similarity functions is provided in Table 5.1. In the table, rows correspond to fusion methods except the first row which is the average of single view query images. It is the case where no fusion methods are applied. The columns are the similarity functions used together with the fusion methods. The time values are in millisecond (ms). In this experiment, five different objects, each with five different images, are used. While calculating the matching time of single image query approach, the average time of all the single image queries is taken. Similarly, for the multi-view query approach, we take the average of the matching time of multi image queries.

Table 5.1: Matching times (ms) of similarity functions using various fusion methods

		Similarity Functions				
		Normalized Correlation	Histogram Intersection	Normalized Histogram Intersection	Dot Product	Max-Min Ratio
Fusion Methods	Single Image Query (No Fusion)	317	343	272	277	332
	Sum Histogram	1122	963	947	899	982
	Average Histogram	911	925	914	902	913
	Maximum Histogram	962	1126	965	988	1010
	Average	1069	1340	877	1242	1132
	Weighted Average	1080	1323	899	1248	1138
	Max	1092	1327	863	1232	1128
	Average Max	1093	1346	858	1243	1136
	Weighted Average Max	1118	1355	849	1299	1130

When we compare the matching times using different similarity functions, we observe that the difference between matching times is not significant. All of the similarity functions requires approximately same amount of time for matching. The difference between the maximum and minimum times is 71 ms. Therefore, the similarity functions can be chosen according to their retrieval performance without considering the time spent during matching. Another point that affects the matching time is the number of query images used in multi image query search. In Table 5.1, first row, shows the average matching time spent for a single query image. The other rows show the matching time spent by various fusion methods which uses five different query images. When the number of images in query set is increased, the matching time also increases as predicted. However, the increase in time is not linear because the number of key points detected in the query images is not constant and the time spent for every query image is different. Although the number of query images is increased from one to five, the increase rate of time spent for matching is not that much. The average time spent for matching is 308 and 1051 for a single query image and fusion methods respectively. This results show that the increase in time is not linearly dependent to the number of query images, which is a desired property for our multi image mobile image search system.

5.3.5 Discussion of the Experimental Results

The experimental results provide a detailed analysis on similarity functions, fusion methods and their effect on matching time. First of all, *Max-Min Ratio* similarity function provides higher average precision compared to other similarity functions. We perform various experiments using single-view query and database images, multi-view query and database images, and query images with and without background clutter. In all of the test environments, *Max-Min Ratio* similarity function results in highest average precision. The reason is that, the function considers both minimum and maximum values of the histogram bins. Therefore, both similar and dissimilar parts between query and database histograms are included in the similarity calculation. It provides a comprehensive calculation which results in more precise similarity value.

Normalized Histogram Intersection and *Normalized Correlation* are the similarity functions which provide the highest precision value after the *Max-Min Ratio* method. One of the reasons for this result is that they are normalized functions which bring the calculated similarity values into a range. Therefore, comparing the similarity values becomes more meaningful. When normalization is not applied, the range of calculated values varies. The difference between two sets of similarity values may mislead the results due to the range difference. Therefore normalized methods perform better compared to not normalized functions. *Dot Product* is one of the methods that does not use normalization. The results prove that the average precision value obtained using this function is low compared to other similarity functions. The reason is that, the large range of similarity values make the comparison unreliable.

Normalized Histogram Intersection and *Normalized Correlation* similarity functions provide similar average precision values when single-view query images are used. Moreover, the background clutter on the query images does not change the relative performance of the similarity functions. In both cases, the average precision of these similarity functions are very close to each other. However, when multi-view query images are used, *Normalized Histogram Intersection* method performs better.

For query images with and without background clutter, the average precision value provided by *Normalized Histogram Intersection* method is higher than *Normalized Correlation*. The extra information provided by multi-view images is utilized better by the *Normalized Histogram Intersection* method, because it takes the minimum normalized value for each histogram bin. This property makes the method more strict. However when the information gathered from the query images is increased, this strictness makes the similarity method perform better.

Dot Product and *Histogram Intersection* similarity methods have lower average precision values compared to other methods. The average precision values provided by these methods are 0.35 and 0.20, which are lower than the other similarity methods when using single-view and multi-view query images, respectively. The difference is smaller when multi-view query images are used because the methods perform better with using extra information gathered from multiple query images. We can conclude that, *Max-Min Ratio* similarity method is a good choice for both single-view and multi-view query image search systems.

The results also provide information about the behavior of the fusion methods. First of all, the increase in average precision value is higher when query images with background clutter are used. The fusion methods increase the average precision value 0.3 and 0.1 for query images with and without background clutter, respectively. The reasons is that, using multi-view images together with fusion methods increases the amount of information gathered from the query images. When background cluttered query images are used, the negative effect of background region is decreased by using provided extra information by multi query images.

For single-view database, the early fusion methods do not perform well when the query images do not have background clutter. However, if the query images are background cluttered, *Average Histogram* and *Maximum Histogram* methods also increase the average precision value.

However, *Sum Histogram* method does not perform well, because summing operation increases the range between the combined histogram and database histogram.

Like the situation with not normalized methods, this summing operation makes the comparison between the histograms unreliable. Taking average or selecting maximum of the values prevents this situation and increases the performance.

Average Histogram fusion method performs better than *Maximum Histogram* method. The reason for this result is that, *Maximum Histogram* methods considers only the similarities between histograms. However, dissimilarity between the histograms is also a source of information. Considering only the similarity of the histogram makes the *Maximum Histogram* method vulnerable to the outliers. If the histogram of the query object is matched to a dissimilar database object with a high similarity value, the method cannot distinguish it. However, the *Average Histogram* method handles this type of objects by taking the average value of histogram bins.

Additionally, *Max* late fusion method also increases the average precision value for query images with background clutter. *Rank Sum* and *Count*, which are late fusion methods, increase the average precision value for query images with and without background clutter. Therefore, these late fusion methods can be used for image search systems if the used database is single-view.

The results prove that ranked based late fusion methods perform better than the others. It is dependent to the quality of the query images. If most of the query images have high quality, like not blurred, then the corresponding results lists have similar objects with good ordering. Therefore, the ranked based methods tolerate the query images that have results lists with dissimilar objects. The query objects of the datasets used in the experiments have good quality, therefore ranked based result lists perform better.

If the query objects may have blurred effect, then the corresponding result list will have dissimilar objects or the similar objects will appear close to the end of the list. In such cases, the *Max* late fusion methods perform better by selecting the result object having the maximum similarity value.

For multi-view database, almost all the early and late fusion methods increase the average precision value independent from the background clutter of the query images. The only exception is *Average* late fusion method for query images having clean background. The result lists of each query object have similar objects with good ordering. In such a case, taking the average will decrease the effect of similar objects by considering the dissimilar ones. The fusion methods perform better compared to using multi-view database, because we also increase the amount of information gathered from database images. Therefore, the methods have more information compared to the case where the single-view database images are used.

Max and *Weighted Average Max* late fusion methods provide the highest average precision values for both types of queries. Therefore, these fusion methods can be used in the image search systems which use multi-view database. Additionally, *Maximum Histogram* early fusion method gives high average precision when the query images have background clutter, because selecting the histogram bins with maximum value decreases the negative effect of the background clutter. As a conclusion, for both types of queries and databases (single-view and multi-view database and query with and without background clutter) late fusion methods perform better than early fusion methods.

The experimental results also provide information about the matching time spent for similarity and fusion methods. The minimum and maximum matching times are 272 ms and 343 ms for *Normalized Histogram Intersection* and *Histogram Intersection*, respectively. The difference is 71 ms that means the matching times of the similarity functions are close to each other. Therefore, the selection of the similarity function can be made without considering the matching time. In general, early fusion methods require less time than late fusion methods for matching. This is reasonable because late fusion methods perform image search for every query image but early fusion methods perform image search once for the combined query histogram. The average matching time difference between early and late fusion methods is 208 ms using queries having 5 different images.

Chapter 6

Conclusion

We propose a mobile image search system that uses a multi image query approach to increase the search performance. To this end, first of all we improve the single image query search approach by combining multiple local features. Then, we apply multi image query search together with fusion methods. We implement and analyze various fusion methods from the literature. In addition to multi image queries, we also apply the multi-view approach to the database images. For this purpose, we provide a new database (*MVOD*) where each object has multiple images representing the different views.

In our experiments, we use different similarity functions to show their effects on performance results by calculating the average precision value. In the results, we observe that the *Max-Min Ratio* and *Normalized Correlation* functions provide the best performance. Additionally, we apply different early and late fusion methods. The results show that late fusion methods, in general, perform better than early fusion methods. In our experiments, we also use queries with and without background clutter to observe its effect. Using multi image queries leads to higher improvement when background cluttered queries are used. The reason for this improvement is that using multi image queries increases the amount of information gathered from the objects, which decreases the negative effect of the background clutter.

We also perform experiments using the *MVOD* database to observe the effect of multi view database images. The results show that using multi view images in the database increases the performance of the image search system. All the early and late fusion methods, except *Average* method, increase the average precision of the image search system. *Max* and *Weighted Average Max* late fusion methods provides the highest average precision values for both types of queries. Additionally, *Maximum Histogram* early fusion method gives high average precision when the query images have background clutter.

We have also examined the matching time of the methods to observe the increase in time when using the multi image query approach together with the multi view database. The results indicate that, the increase in time is not linear with the number of query images. The reason is that, the key points detected for each query image is not constant. The increase in time is less than the linear increase. For similarity methods, the matching time is close to each other. This is also valid for the matching time spent when using fusion methods. The matching time of the fusion methods is close to each other which allows us to select the fusion methods according to their retrieval performance.

Several extensions can be considered for further improvement in the performance of our image search system. Firstly, the number of images in the *MVOD* database can be increased to have better representation of the stored objects. Moreover, the similarity and fusion methods can be selected dynamically according to the average precision of the single image queries. Additionally, different similarity methods can be proposed to calculate the similarity between the query and the database.

Bibliography

[1] B. Girod, V. Chandrasekhar, R. Grzeszczuk, and Y. A. Reznik, "Mobile visual search: Architectures, technologies, and the emerging MPEG standard," *IEEE MultiMedia*, vol. 18, no. 3, pp. 86–94, 2011.

[2] X. Shen, Z. Lin, J. Brandt, and Y. Wu, "Mobile Product Image Search by Automatic Query Object Extraction," in *Proceedings of the European Conference on Computer Vision*, pp. 114–127, 2012.

[3] M. Bastan, H. Cam, U. Gudukbay, and O. Ulusoy, "Bilvideo-7: An MPEG-7-compatible video indexing and retrieval system," *IEEE Multimedia*, vol. 17, no. 3, pp. 62–73, 2010.

[4] S. Zhang, M. Yang, T. Cour, K. Yu, and D. N. Metaxas, "Query specific fusion for image retrieval," in *ECCV (2)*, vol. 7573 of *Lecture Notes in Computer Science*, pp. 660–673, 2012.

[5] U. Niaz and B. Merialdo, "Fusion methods for multimodal indexing of web data," in *International Workshop on Image and Audio Analysis for Multimedia Interactive Services*, (Paris, France), 2013.

[6] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma, "A Survey of Content-based Image Retrieval with High-level Semantics," *Pattern Recognition*, vol. 40, no. 1, pp. 262–282, 2007.

[7] P. Nimi and C. Tripti, "Feature based image retrieval algorithm," in *Advances in Computing and Communications*, pp. 46–55, Springer Berlin Heidelberg, 2011.

[8] S. Joseph and K. Balakrishnan, "Multi-Query Content Based Image Retrieval System using Local Binary Patterns," *International Journal of Computer Applications*, vol. 17, pp. 1–5, March 2011.

[9] A. N. Bhute and B. B. Meshram, "Content Based Image Indexing and Retrieval," *CoRR*, vol. abs/1401.1742, 2014.

[10] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by Image and Video Content: The QBIC System," *Computer*, vol. 28, September 1995.

[11] A. Pentland, R. W. Picard, and S. Sclaroff, "Photobook: Content-based manipulation of image databases," *International Journal of Computer Vision*, vol. 18, no. 3, pp. 233–254, 1996.

[12] J. R. Smith and S.-F. Chang, "VisualSEEk: A Fully Automated Content-based Image Query System," in *Proceedings of the ACM International Conference on Multimedia*, pp. 87–98, 1996.

[13] J. Z. Wang, G. Wiederhold, O. Firschein, and S. Xin Wei, "Content-based image indexing and searching using daubechies' wavelets," *International Journal on Digital Libraries*, vol. 1, no. 4, pp. 311–328, 1998.

[14] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik, "Blobworld: a system for region-based image indexing and retrieval," Tech. Rep. UCB/CSD-99-1041, EECS Department, University of California, Berkeley, 1999.

[15] C. H. Lampert, "Detecting objects in large image collections and videos by efficient subimage retrieval," in *Proceedings of the IEEE 12th International Conference on Computer Vision*, pp. 987–994, Sept 2009.

[16] B. Szanto, P. Pozsegovics, Z. Vamossy, and S. Sergyan, "Sketch4match–content-based image retrieval system using sketches," in *Proceedings of the IEEE 9th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pp. 183–188, Jan 2011.

[17] R. M. Mohana and A. R. M. Reddy, “CCBIR: A cloud based implementation of content based image retrieval,” *WSEAS Transactions on Computers*, vol. 14, 2015.

[18] L. Liu, M. Yu, and L. Shao, “Multiview alignment hashing for efficient image search,” *IEEE Transactions on Image Processing*, vol. 24, pp. 956–966, March 2015.

[19] X. Xie, L. Lu, M. Jia, H. Li, F. Seide, and W.-Y. Ma, “Mobile search with multimodal queries,” *Proceedings of the IEEE*, vol. 96, pp. 589–601, April 2008.

[20] S. Madhu, “Content based image retrieval: A quantitative comparison between query by color and query by texture,” *Journal of Industrial and Intelligent Information*, vol. 2, no. 2, pp. 108–112, 2014.

[21] B. Girod, V. Chandrasekhar, D. M. Chen, N. Cheung, R. Grzeszczuk, Y. A. Reznik, G. Takacs, S. S. Tsai, and R. Vedantham, “Mobile Visual Search,” *IEEE Signal Processing Magazine*, vol. 28, no. 4, pp. 61–76, 2011.

[22] K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1615–1630, Oct 2005.

[23] T. Tuytelaars and K. Mikolajczyk, “Local invariant feature detectors: A survey,” *Foundations and Trends in Computer Graphics and Vision*, vol. 3, no. 3, pp. 177–280, 2008.

[24] Y. Su, T. Chiu, Y. Chen, C. Yeh, and W. H. Hsu, “Enabling low bitrate mobile visual recognition: a performance versus bandwidth evaluation,” in *ACM Multimedia Conference, MM ’13, Barcelona, Spain, October 21-25, 2013*, pp. 73–82, 2013.

[25] J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo, “Evaluating bag-of-visual-words representations in scene classification,” in *Proceedings of the International Workshop on Multimedia Information Retrieval*, MIR ’07, pp. 197–206, 2007.

[26] V. Chandrasekhar, D. M. Chen, A. Lin, G. Takacs, S. S. Tsai, N.-M. Cheung, Y. A. Reznik, R. Grzeszczuk, and B. Girod, "Comparison of local feature descriptors for mobile visual search," in *IEEE International Conference on Image Processing*, pp. 3885–3888, 2010.

[27] L. Duan, F. Gao, J. Chen, J. Lin, and T. Huang, "Compact descriptors for mobile visual search and MPEG CDVS standardization," in *IEEE International Symposium on Circuits and Systems*, pp. 885–888, 2013.

[28] J. S. Hare and P. H. Lewis, "Content-based image retrieval using a mobile device as a novel interface," in *Storage and Retrieval Methods and Applications for Multimedia*, vol. SPIE V, pp. 64–75, SPIE and IS&T, 2005.

[29] M. Noda, H. Sonobe, S. Takagi, and F. Yoshimoto, "Cosmos: Convenient image retrieval system of flowers for mobile computing situations," in *Proceedings of the IASTED International Conference Information Systems and Databases (ISDB 2002)*, pp. 25–30, 2002.

[30] H. Sonobe, S. Takagi, and F. Yoshimoto, "Image Retrieval System of Fishes Using a Mobile Device," in *Proceedings of International Workshop on Advanced Image Technology*, pp. 33–37, 2004.

[31] T. Yeh, K. Grauman, K. Tollmar, and T. Darrell, "A picture is worth a thousand keywords: image-based object search on a mobile platform," in *Proceedings of the Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pp. 2025–2028, 2005.

[32] D. M. Chen and B. Girod, "Memory-Efficient Image Databases for Mobile Visual Search," in *IEEE Multimedia*, vol. 21, pp. 14–23, 2013.

[33] Y. Wang, T. Mei, J. Wang, H. Li, and S. Li, "JIGSAW: Interactive Mobile Visual Search with Multimodal Queries," in *Proceedings of the 19th ACM International Conference on Multimedia*, MM '11, pp. 73–82, 2011.

[34] D. Li and M. C. Chuah, "Emod: An efficient on-device mobile visual search system," in *Proceedings of the 6th ACM Multimedia Systems Conference*, pp. 25–36, 2015.

[35] T. Guan, Y. He, L. Duan, J. Yang, J. Gao, and J. Yu, "Efficient bof generation and compression for on-device mobile visual location recognition," *IEEE Multimedia*, vol. 21, no. 2, pp. 32–41, 2014.

[36] J. Mennesson, P. Tirilly, and J. Martinet, "Elementary block extraction for mobile image search," in *International Conference on Image Processing* (IEEE, ed.), (Paris, France), 2014.

[37] R. Ji, L. Duan, J. Chen, T. Huang, and W. Gao, "Mining compact bag-of-patterns for low bit rate mobile visual search," *IEEE Transactions on Image Processing*, vol. 23, no. 7, pp. 3099–3113, 2014.

[38] W. Zhou, M. Yang, H. Li, X. Wang, Y. Lin, and Q. Tian, "Towards codebook-free: Scalable cascaded hashing for mobile image search," *IEEE Transactions on Multimedia*, vol. 16, no. 3, pp. 601–611, 2014.

[39] Google Inc., "Google Goggles." `http://www.google.com/mobile/goggles`, 2015. Accessed: 2015-05-29.

[40] X. Liu, J. J. Hull, J. Graham, J. Moraleda, and T. Bailloeul, "Mobile visual search, linking printed documents to digital media," 2010.

[41] Nokia Inc., "Nokia Point and Find." `http://www.pointandfind.nokia.com`, 2015. Accessed: 2015-05-29.

[42] Qualcomm Connected Experiences, Inc., "Kooaba: Image Recognition." `http://www.kooaba.com`, 2015. Accessed: 2015-06-12.

[43] J. Graham and J. J. Hull, "iCandy: a Tangible User Interface for iTunes," in *Proceedings of the Conference on Human Factors in Computing Systems*, pp. 2343–2348, 2008.

[44] SnapTell, Inc., "SnapTell." `http://www.snaptell.com`, 2015. Accessed: 2015-05-29.

[45] Digimarc, Inc., "Digimarc Discover." `http://www.digimarc.com/discover`, 2015. Accessed: 2015-05-29.

[46] David M. Chen, "Mobile Visual Search." http://web.stanford.edu/~dmchen/mvs.html, 2015. Accessed: 2015-05-29.

[47] AndroidTapp, Inc., "PlinkArt." http://www.androidtapp.com/plinkart, 2015. Accessed: 2015-05-29.

[48] Image Searcher, Inc., "CamFind." http://camfindapp.com, 2015. Accessed: 2015-05-29.

[49] R. G. Salai, "CCBIR: A cloud based implementation of content based image retrieval," *Journal of Theoretical and Applied Information Technology*, vol. 74, no. 2, 2015.

[50] Amazon Inc., "Flow Powered by Amazon." http://www.a9.com/whatwedo/mobile-technology/flow-powered-by-amazon. Accessed: 2015-07-02.

[51] N. Zhang, T. Mei, X.-S. Hua, L. Guan, and S. Li, "Taptell: Interactive visual search for mobile task recommendation," *Journal of Visual Communication and Image Representation*, vol. 29, no. 0, pp. 114 – 124, 2015.

[52] R. Arandjelovic and A. Zisserman, "Multiple queries for large scale specific object retrieval," in *Proceedings of the British Machine Vision Conference*, pp. 92.1–92.11, BMVA Press, 2012.

[53] B. Fernando and T. Tuytelaars, "Mining Multiple Queries for Image Retrieval: On-the-fly learning of an Object-specific Mid-level Representation," 2013.

[54] C.-H. Lee and M.-F. Lin, "A Multi-query Strategy for Content-based Image Retrieval," *International Journal of Advanced Information Technologies*, vol. 5, no. 2, pp. 266–275, 2012.

[55] M. Mazloom, A. H. Habibian, and C. G. M. Snoek, "Querying for video events by semantic signatures from few examples," in *Proceedings of the ACM Multimedia Conference, MM '13, Barcelona, Spain*, pp. 609–612, 2013.

[56] J. Tang and S. Acton, "An image retrieval algorithm using multiple query images," in *Proceedings of the Seventh International Symposium on Signal Processing and Its Applications*, vol. 1, pp. 193–196, July 2003.

[57] B. Moghaddam, H. Biermann, and D. Margaritis, “Regions-of-interest and spatial layout for content-based image retrieval,” *Multimedia Tools and Applications*, vol. 14, no. 2, pp. 201–210, 2001.

[58] C. Zhang, X. Chen, and W.-B. Chen, “An Online Multiple Instance Learning System for Semantic Image Retrieval,” in *IEEE International Symposium on Multimedia Workshops*, pp. 83–84, Dec 2007.

[59] Y. Xue, X. Qian, and B. Zhang, “Mobile image retrieval using multi-photos as query,” in *Proceedings of the IEEE International Conference on Multimedia and Expo Workshops*, pp. 1–4, July 2013.

[60] D. A. Lisin, M. A. Mattar, M. B. Blaschko, E. G. Learned-Miller, and M. C. Benfield, “Combining local and global image features for object class recognition,” in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, 2005.

[61] T. Tuytelaars and K. Mikolajczyk, “Local invariant feature detectors: A survey,” *Foundations and Trends in Computer Graphics and Vision*, pp. 177–280, 2008.

[62] J. Chen, L. hui Zou, J. Zhang, and L. hua Dou, “The comparison and application of corner detection algorithms,” *Journal of Multimedia*, vol. 4, no. 6, pp. 435–441, 2009.

[63] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded Up Robust Features,” in *Proceedings of European Conference on Computer Vision, ECCV 2006*, Lecture Notes in Computer Science, vol. 3951, pp. 404–417, Springer Berlin Heidelberg, 2006.

[64] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, “Lost in quantization: Improving particular object retrieval in large scale image databases,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*, 2008.

[65] F.-F. Li and P. Perona, "A Bayesian hierarchical model for learning natural scene categories," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, (Washington, DC, USA), pp. 524–531, IEEE Computer Society, 2005.

[66] S. seok Choi and S. hyuk Cha, "A survey of binary similarity and distance measures," *Journal of Systemics, Cybernetics and Informatics*, pp. 43–48, 2010.

[67] S.-H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 1, no. 4, pp. 300–307, 2007.

[68] L. Zhu and A. Zhang, "Supporting Multi-Example Image Queries in Image Databases," in *Proceedings of the International Conference on Multimedia and Expo*, pp. 697–700, 2000.

[69] A. Gunawardana and G. Shani, "A survey of accuracy evaluation metrics of recommendation tasks," *Journal of Machine Learning Research*, vol. 10, pp. 2935–2962, Dec. 2009.

[70] G. Griffin, A. Holub, and P. Perona, "Caltech-256 Object Category Dataset," Tech. Rep. CNS-TR-2007-001, California Institute of Technology, 2007.

[71] "OpenCV." `http://opencv.org`, 2015. Accessed: 2015-07-27.