INSTANCE RETRIEVAL AND IMAGE AUTO-ANNOTATIONS ON MOBILE DEVICES

Thesis submitted in partial fulfillment
of the requirements for the degree of

Dual Degree (BTech + MS by Research)
in
Computer Science

by

JAYAGURU PANDA
200802017
jayaguru.panda@research.iiit.ac.in

Center for Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
December 2013

Copyright © Jayaguru Panda, 2013
All Rights Reserved

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "INSTANCE RETRIEVAL AND IMAGE AUTO-ANNOTATIONS ON MOBILE DEVICES" by Jayaguru Panda, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Adviser: Prof. C V JAWAHAR

To my beloved Mother

Smt. Nirmala Panda

Acknowledgments

First of all, I owe my deepest thanks and gratitude to my adviser, Prof Dr. C. V. Jawahar, for his unparalleled support, motivation, dedication, time and encouragement. His guidance has taught me a lot of things both on academic and personal levels. I am immensely inspired from his work ethics and his attitude towards looking at any problem and I consider myself blessed and grateful for the times we had deep thoughtful discussions. His vision gave me an opportunity to work on an emerging problem in the field of computer vision on mobile platforms and eventually led to publications at top-tier conferences. Many thanks to Prof Dr. Michael S. Brown (of NUS, Singapore), without whom, this work would not have been the same. I would also like to thank my friend and batchmate Shashank with whom I started my journey at CVIT research center.

I am thankful to my lab mates and CVIT peers for having all the fruitful and useful discussions, and for the sleepless nights we were working together before deadlines Srijan, Mayank, Abhinav, Vinay, Siddhartha, Ankit, Chandrashekar, Harshit, Nitish, Anand, Mihir, Parikshit, Aditya, Praveen. I would also like to thank all my friends at IIIT - Antarip, Aman Jain, Saxena, Naveen, Manushree, Sandhya, Baheti, Mudit, Swapnil, Daga, Abhishek, Pranav, Ankit Agarwal, Zade, Anusha, Divya for supporting me through good and bad times, and making life at (and after) IIIT a remarkable one.

A special thanks to Dr. P. J. Narayanan, Dr. Anoop Namboodiri, Dr. Jayanthi Sivaswamy for creating a phenomenal environment in CVIT to do research and CVIT administration Satya, Phani, Rajan for helping me on numerous occasions.

Last but denitely above all, I thank my parents, my brother and all my family for their constant love, support and belief. I would like to specially acknowledge my mother, who has always played a very encouraging and supporting role in all my endeavours. I shall be forever indebted to my relatives and cousins who have put their faith in me all these years and blessed me in every sphere of life.

Abstract

Image matching is a well studied problem in the computer vision community. Starting from template matching techniques, the methods have evolved to achieve robust scale, rotation and translation invariant matching between two similar images. To this end, people have chosen to represent images in the form of a set of descriptors extracted at salient local regions that are detected in a robust, invariant and repeatable manner. For efficient matching, a global descriptor for the image is computed either by quantizing the feature space of local descriptors or using separate techniques to extract global image features. With this, effective indexing mechanisms are employed to perform efficient retrieval on large image databases.

Successful systems have been put in place in desktop and cloud environments to enable image search and retrieval. The retrieval takes fraction of a second on a powerful desktop or a server. However, such techniques are typically not well suited for less powerful computing devices such as mobile phones or tablets. These devices have small storage capacity and the memory usage is also limited. Computer vision algorithms run slower, even when optimized for the architecture of mobile processors. These handheld devices, or so-called smart devices are increasingly used for simple tasks that seem too trivial for a desktop or a laptop and can be easily accessed on a smaller display. Further, they are more popularly used for taking pictures (gradually replacing the space of digital cameras) owing to the improved embedded camera sensors. Hence, a user is more likely to use a query image from the mobile phone, rather than from the desktop. This increases the scope of applications that demand real-time search and retrieval result delivered on a mobile phone.

Many applications (or apps) on mobile smart phones communicate with the cloud to perform tasks that are infeasible on the device. People have attempted to retrieve images in this cloud-based model by either sending the image or its features to the server and receiving back relevant information. We are interested to solve this problem on the device itself with all the necessary computations happening on the mobile processor. It allows a user to not bother for a consistent network connection and the communication overheads associated with the search process. We address the range of applications that need simple text annotations to describe the image queried on the mobile. An interesting use case is a tourist/student/historian visiting a heritage site and can get all information about the monuments and structures on his mobile phone. Once the app is initialized on the device, the camera is opened and just pointing the camera or with a single click all the useful info about the monument is displayed on the screen instantly. The app doesn't use the internet for communicating with any server and should do

all computations on the mobile phone itself. Our methods optimize the process of instance retrieval to enable quick and light-weight processing on a mobile phone or a tablet.

Firstly, we obtain a dataset of images relevant to the application's scope, which is then curated and annotated for the task. This may seem a trivial task, but involves a great deal of manual labour when dealing with datasets of thousands of images. We efficiently address this with a batch annotation approach that results in detailed and part-wise image annotations. Then, we look into the primary problem of performing image retrieval on a mobile phone and propose variants of the Bag-of-words approach. A new indexing mechanism helps us to reduce the memory footprint by a large extent. It also reduces the computational requirements. Finally, we develop the mobile app using existing computer vision libraries for smart phones, keeping in mind a simple and user-friendly design for seamless use by a wide range of users.

We validate our approach by using the standard benchmark datasets of Oxford-5K for our experimental purposes and also show results on Paris-5K and UKBench datasets. We demonstrate a working mobile app designed for a heritage site that successfully retrieves relevant information about a monument captured by the mobile phone.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Consider teaching a kid about the Taj Mahal. The first thing one does is show the photo of the monument and tell him, "This is the Taj Mahal". The human brain is designed to seamlessly work with the visual system so as to be able to recognize patterns and learn from experience. The kid quickly registers the photo of this uniquely beautiful structure in his brain's memory. The next time he sees a similar photo of the Taj Mahal, he quickly resembles it with the one seen before and gathers up all the facts learnt about it. We would like to arm a computing device with similar cognitive ability, which can then be trained for various tasks. However, a computer or a machine captures visual information in the form of a digital image, which is a set of raw pixel values. The fields of computer vision and machine learning deal with analysing raw digital data and abstract high-level semantic information. In short, the techniques are targeted to power a machine with human-like perception and self-learning abilities. To this end, the computer must be able to remember a digital photo of the Taj Mahal and recollect the information learnt when it is given a similar photo later.

In this work, we focus on using low-end computing devices like mobile phones and tablets that have limited processing and memory capacities to adapt computation-intensive vision algorithms. Successful computer vision algorithms that run on powerful desktop or server platforms fail to work in such environments. We are primarily motivated by the problem of on-spot recognition of heritage monuments and auto-annotations with rich textual information for the use of tourists, historians, students and enthusiasts alike.

1.1 Recognition tasks in Computer Vision

Visual recognition essentially poses the following questions with respect to the information present in an image: a) What object(s) are present ? b) Where are they present? c) Is this similar to any other image in the database? Computer vision algorithms try to give a convincing answer to such questions by analysing the various scenarios where such questions are posed. Subsequently, it is realised that the set of recognition problems can be divided into the following types and must be solved within its own merit:

1.1.1 Detection

This problem addresses the task of detecting and localizing semantic objects belonging to a particular category. For example, a digital image needs to be processed to check whether it contains a person and if so, which part of the image has the person. The solution can be in the form of a tight rectangular bounding box around the object in the image. The popular approach is to use the sliding window method and check each candidate rectangular bounding window within the image.

1.1.2 Classification

Image classification is a traditional problem in computer vision dealing with the category of the object captured in the image. An object category includes similar looking objects grouped together semantically. For example all chairs are grouped into one category, while all beds are grouped into a separate category. Now, given an image, the problem is to identify whether it contains a chair or a bed. Typically, there are a predefined set of candidate categories, for which we have sample images to teach or train a computer. The algorithm must predict the most appropriate category for a new image. The solution typically involves using a labelled training set of objects to learn the visual patterns and features that distinguishes the image of a specific object category from the rest and build classifiers based on this. A new image is then processed and classified into one of the categories.

1.1.3 Matching

Image matching is another classical problem in computer vision that compares the images of a specific object instance taken in different conditions like dissimilar background, presence of other occluding objects, and view point and view angle variations. The task is to robustly match two distinct images and predict whether they are contain the same object instance. People have tried various methods for this task, starting from the template matching approach. More popularly, local invariant features are extracted at salient regions in the images and then, compared for two given images. The algorithm gives a matching score that determines the probability of the two images belonging to a specific object.

1.1.4 Retrieval

This problem addresses a document-retrieval analogue for large image datasets, where a query image is looked up for similar images. The computer vision community works on two kinds of retrieval problems (See Figure 1.1) (i) Category Retrieval or, object class recognition, which concerns with the retrieval of a category/class of objects. For example, given a query of the image for a house, it retrieves images of all variations of houses like multi-storeyed, duplex, bungalows, as well as cottages. The solution builds upon that of the Classification problem and designed to work with large number of categories. (ii) Instance, or object retrieval, which deals with retrieval of a particular object. Here, when the query is an image of The White House, the system must only retrieve images of The White House,

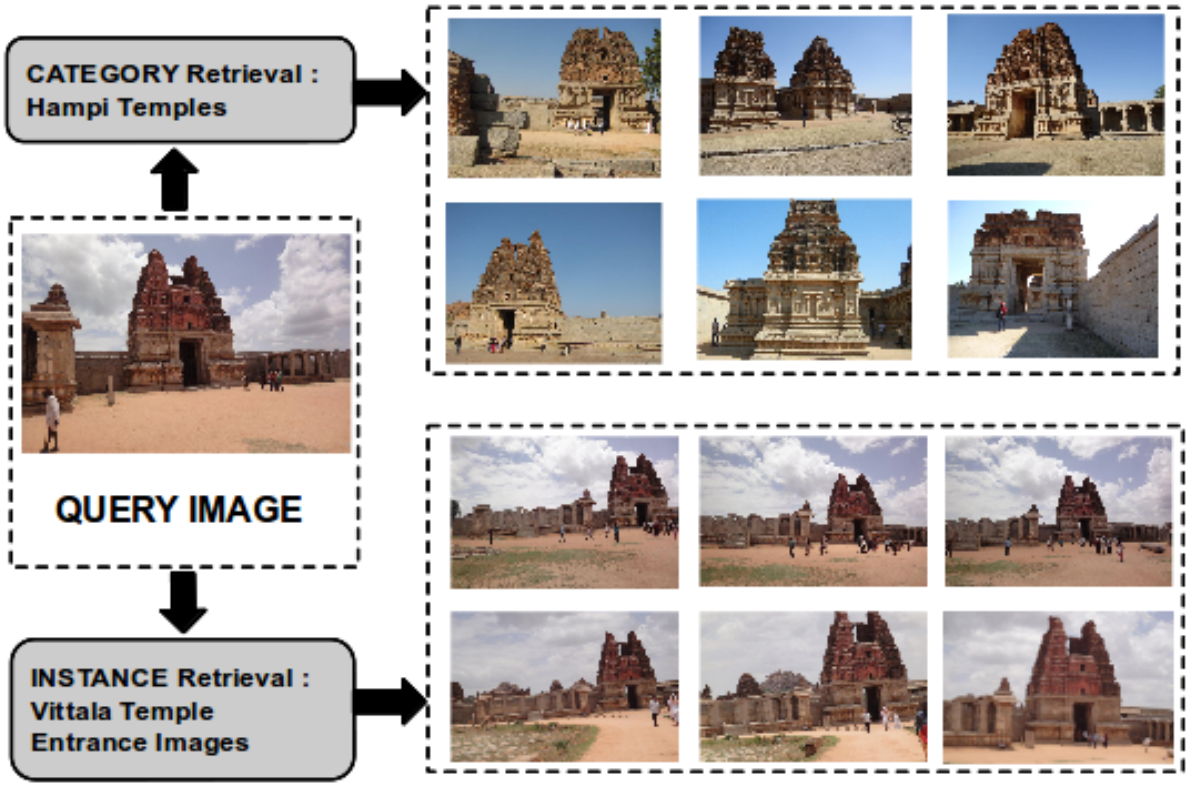


Figure 1.1: Category Retrieval Vs Instance Retrieval: Given a query image of a monument, Category retrieval addresses the problem of retrieving images of similar looking monuments that may share similar architectural styles. While Instance retrieval seeks to retrieve images of the same monument captured with wide viewing variations.

captured in varied viewing conditions but sharing structural overlaps with the query. The solution for this builds upon the Matching problem, when it involves matching a query image with a large dataset of images. It uses efficient indexing techniques for faster search and retrieval.

1.2 Mobile Phones and Computer Vision Apps

Classical computer vision and machine learning algorithms are developed and evaluated on high-end computing servers. They either get live input from digital cameras (for example, surveillance system servers) or are given pre-captured photos/videos as inputs (for example, uploading on a web-based system). Pre-captured photos may come from common digital cameras, scanned print photos or more recently from mobile phone cameras. These days standard mobile phones (or, smartphones) have decent embedded digital camera sensors, strong wireless communication support for Internet connectivity and a mobile processing unit. It is easy and intuitive to take pictures and view them on these handheld devices.

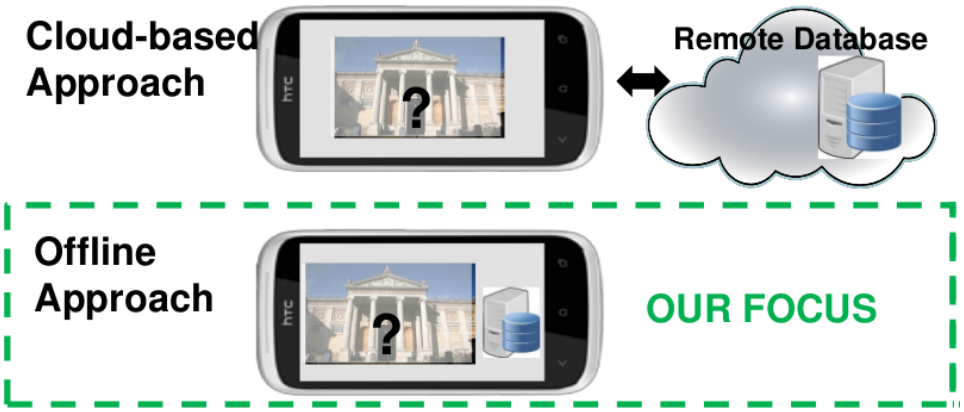


Figure 1.2: A monument recognition task involves computation intensive vision algorithms to process the query image and extract high-level semantic information. This can be accomplished on a mobile phone in two ways: (a) Cloud-based approach, where an internet connectivity is required to communicate with a powerful remote server. (b) Offline approach, where the algorithm is optimized to run on an offline device and the entire database/index has affordable memory footprint.

Moreover, the emphasis on improved camera sensors in mobile phones is constantly increasing keeping in view the large target audience ready to use their phones as a replacement to digital cameras. Such an environment makes mobile platforms the ideal choice to run real-time computer vision tasks and develop interesting applications. However, existing computer vision solutions are designed to run on high-end desktop or server computers that have huge storage, memory (RAM) capacity and powerful processing units. From this perspective, a mobile phone platform is constrained for running low-end tasks that can work in limited memory and processing conditions. This is addressed in two ways (See Figure 1.2).

1.2.1 Cloud-based apps

This involves performing minimal processing on the mobile device, while the real processing (heavy computation jobs) is handled at a remote server end, where powerful computers are assigned the task. The mobile phone communicates with the server (or cloud) to accomplish the task with the help of wifi or mobile internet data connections. The image or its features (usually in compressed form) is sent over to the server, where the actual processing takes place. The annotation retrieved is sent back to the mobile. A lot of commercial mobile apps that deliver recognition solutions are based on this model.

This client-server model suffers a few disadvantages. In the absence of a consistent wireless internet connection the mobile phone is useless at the task. Further there is a need for considerable bandwidth to transfer the images or its features to the server and such services may turn out to be too expensive.

Figure 1.3: A typical use case for the problem posed in this thesis. A user captures a photo on his mobile phone at a heritage site and gets instant auto-annotations displayed on the screen of the phone.

1.2.2 Standalone Offline apps

There is a rising need that useful applications that need to run heavy computer vision algorithms must be optimized to use the minimal processing capabilities of a mobile phone and performed in a standalone or offline mode. The memory (RAM) of a mobile phone is limited to 1-2GB which is partially available for a mobile app. Mobile phones have up to 4-32GB of storage, which is mainly in the form of slower flash memory that is an order of magnitude slower than RAM. The memory footprint of the resident app must be within tens of MBs to make it possible to fit in the RAM. This thesis solves the problem of performing a recognition task as a standalone mobile app within the constraints of a low-end mobile phone.

1.3 Problem Statement and Goal of thesis

We are interested to work with images of monuments and architectural structures captured at cultural heritage sites. Specifically, given a query image of a monument, our system must match it against the dataset and retrieve relevant information about it. We target mobile handheld devices as the primary scope for this sort of application, as they are intuitive to use for clicking photos at a tourist or heritage

site and look-up information about the monument. The use case can be pictured as a tourist or a student visiting a heritage site and interested in specific details of the structures. He has a mobile phone to take photos and fantasizes all useful information about the monument captured to be automatically overlaid or displayed along with the photo on his mobile phone. This requires the mobile phone to instantly auto-annotate the captured photo by recognizing the monument or structure in it and displaying relevant details about it (See Figure1.3).

In order to recognize the monument in a query image, we need to find the best match in the annotated dataset of images for the heritage site. The basic solution is to perform instance retrieval on a database of annotated images and rank the list of most similar images. The image on top of the ranked list (after verification) can be assumed to contain the same monument as in the query image. This is now the annotation source, and all details attached to the source fits the description of the query image. We are interested to work with offline mobile platforms where all computations must happen on the device without the need to communicate with a remote server or the cloud.

1.4 Instance Retrieval with Bag of Words (BoW)

There are a lot of approaches for Instance retrieval that use a global representation of each image, which can be compared with that of the query image. One of the popular approaches is to use the Bag of Words (BoW) model followed by building an effective search index for efficient retrieval.

1.4.1 Bag of Words (BoW) model

This is a simplifying representation used in the context of Information Retrieval (IR) and classification. It was first used for representing text (a sentence, paragraph or a document) in the form of a bag or a multiset, disregarding grammar or even the word order. The multiplicity of the words occurring in the document is maintained in the model so that the frequency of a word is used as a feature for representation. The text document is represented as a histogram of words and a term weighting scheme is employed for both IR and text classification tasks. The most popular term weighting scheme is the TF-IDF (Term Frequency-Inverse Document Frequency), which is a numerical statistic to reflect how important a word is to a document in a database or a corpus.

In Computer Vision, the BoW (also known as Bag of Visual Words) model, first introduced in [44], draws its parallels by using salient interest regions in an image and treating similar regions as the same "visual word" (See Figure 1.4). Robust and invariant local feature points are detected at salient image regions and transformation invariant feature descriptors are extracted at those points. Popular detectors used for this purpose are SIFT, harris-affine, hessian-affine, MSER, etc. Popular descriptor extractors are SIFT, SURF, HOG, ORB, etc. These descriptors are typically high dimensional fixed sized vectors. Visual words are defined for such descriptors as a grouping of set of descriptors that occur close by in the high-dimensional feature space. A visual word vocabulary is obtained by performing a clustering

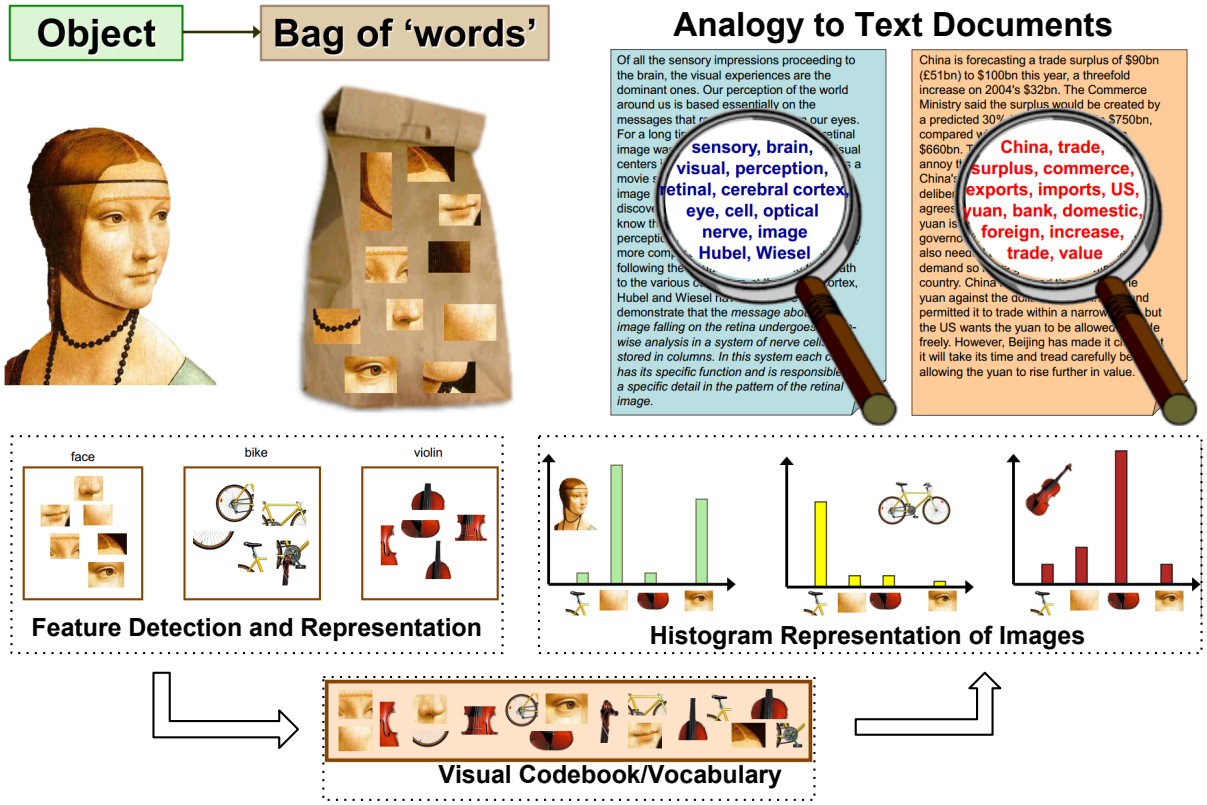


Figure 1.4: A graphical representation of Bag of Words (BoW) model, that shows how an image of a object is represented as a bag or multiset of visual words. Its analogy with text documents is also clear. In the example, 3 objects (face, bike and violin) can be seen to be intuitively composed of their local interest regions and a histogram of visual word vocabulary (or the parts) is used to represent them.

(like K-Means) in the feature space for a set of image descriptors from all images in the dataset. As the feature descriptors are clustered, each descriptor is now assigned a cluster ID corresponding to the cluster it belongs to. The cluster itself is represented as a vector which is the mean of all descriptors assigned to it. Each image can now be represented as a Vocabulary-sized histogram of visual words that captures the frequency of occurrence of each visual word in it.

1.4.2 Inverted Search Index

The instance retrieval problem needs matching the query image against the entire database. Using the BoW model, we have obtained a compact numerical representation of each image as Histogram of visual words. During retrieval, local feature descriptors are extracted at interest points from the query image and assigned to the respective clusters corresponding to the existing visual word vocabulary. The query image is now represented as a BoW histogram. This histogram must be compared with the histograms

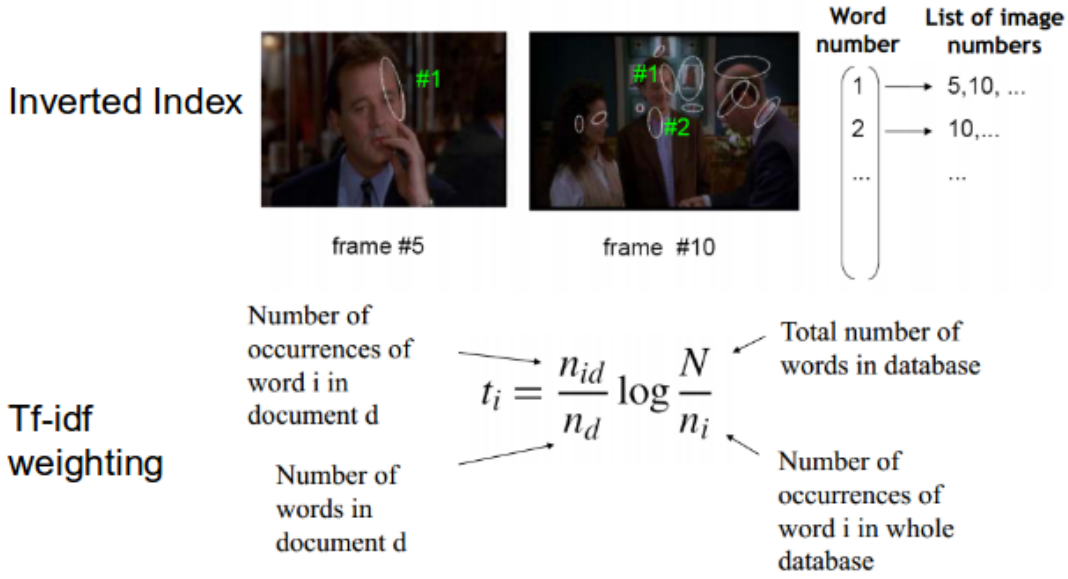


Figure 1.5: Inorder to make the online query process efficient, Bag-of-Words (BoW) representation for each image is used to build the Inverted Index, where each visual word in the vocabulary (or codebook), is indexed to a list of database images it occurs in. The TF-IDF weighting measure used to score the images during online query retrieval is also depicted here.

of each database image to find the closest matches. This is posed as a Nearest Neighbour problem and histograms of two images are compared by a cosine dot product metric. For efficient retrieval, an inverted index is built where each visual word in the vocabulary is indexed to a posting list of database images they occur in. Again during the retrieval phase, each visual word occurring in the query, is looked up against its posting list and the database images are scored accordingly with the number of common visual features they share with the query. A TF-IDF (Term Frequency-Inverse Document Frequency) weighting measure is usually employed to improve the scoring process (See Figure 1.5). A list of images ranked in order of decreasing scores is finally retrieved.

1.4.3 Matching for Geometric Verification

The BoW model fails to incorporate the spatial information of features in the image representation. The retrieved images are scored on basis of the common visual word features in the query image, without zero emphasis on the spatial and/or geometric consistency of the matched or common features. Hence, the retrieved list must be verified to confirm the correctness of similarity the the query. For this, the high-dimensional descriptors in the query and target images are matched similar to David Lowe's SIFT matching method [33]. This gives the feature point correspondences between the query and a retrieved image. A fundamental matrix fit is then estimated between both the images using the 7-point RANSAC algorithm and the number of inlier matches gives a matching score (See Figure 1.6). The retrieved

Figure 1.6: An example of spatial verification on Hampi images, where a Fundamental Matrix is fit across two images.

images are then re-ranked based on the matching scores. This post-processing step typically improves the mean Average Precision (mAP) of Instance Retrieval [38].

1.4.4 Performance Evaluation

The standard measure of mean Average Precision (mAP) to evaluate the performance in Information Retrieval approaches is used in instance retrieval frameworks as well. Precision, recall, and the F measure are set-based measures. They are computed using unordered sets of documents. We need to extend these measures (or to define new measures) if we are to evaluate the ranked retrieval results that are now standard with search engines. In a ranked retrieval context, appropriate sets of retrieved documents are naturally given by the top K retrieved documents. For each such set, precision and recall values can be plotted to give a precision-recall curve , such as the one shown in Figure 1.7. Precision-recall curves have a distinctive saw-tooth shape: if the $(K+1)^{th}$ document retrieved is nonrelevant then recall is the same as for the top K documents, but precision has dropped. If it is relevant, then both precision and recall increase, and the curve jags up and to the right.

In the context of instance retrieval, Precision is dened as the ratio of retrieved positive images to the total number retrieved. Recall is dened as the ratio of the number of retrieved positive images to the

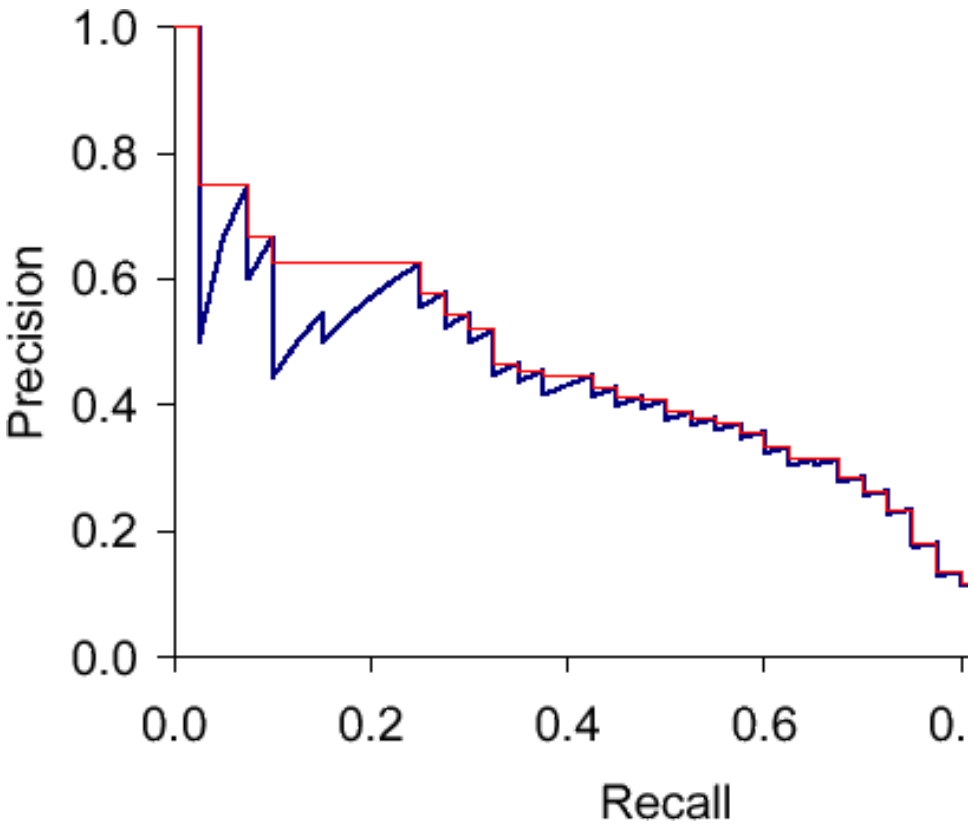


Figure 1.7: A typical Precision-Recall curve showing a distinctive saw-tooth shape. The area under the curve is the Average Precision of the retrieval method on a query. An ideal curve has precision 1 over all recall levels and this corresponds to an Average Precision (AP) of 1.

total number of positive images in the corpus. An ideal precision-recall curve has precision 1 over all recall levels and this corresponds to an average precision of 1 [38]. To evaluate a retrieval approach, an average precision score is obtained for each of the images used as query. Averaging these scores over all queries for a particular dataset, gives the mean Average Precision (mAP). The average of these mAP scores is used as a single number to evaluate the overall performance of an approach on a particular dataset.

1.4.5 Challenges for a Mobile Phone

Using an inverted search index with the BoW model, retrieval can in fact be done very efficiently, say in sub seconds from millions of images on a typical desktop. However, the RAM and storage requirements are high for a mobile phone. Further, the geometric verification step involves matching the high dimensional descriptors in query image with those in the target image to obtain a minimal number of correspondences, which then helps in generating a transformation hypothesis. To store the

high-dimensional descriptors on a mobile phone, demands considerable Internal or SD-Card storage space. The matching step is also computationally heavy since a large number of high-dimensional descriptors are compared between a query image and each of the retrieved images. This makes the Geometric Verification step too slow to return instantaneous results on a mobile phone.

1.5 Focus and Organization of Thesis

As realized in the previous section, the existing Instance Retrieval pipeline with the BoW framework is not well-suited to run on a mobile phone. This thesis strives to solve the problem and deliver a robust mobile application, designed to successfully recognize monuments and instantly auto-annotate them using an offline device. The focus is to develop a mobile instance retrieval framework that retrieves the true match at the top of the list, from an annotated dataset of images specific to a heritage or tourist location. The following briefly explains the organization of the rest of the thesis:

- Chapter 2 proposes an efficient batch annotation approach to effectively build a curated and annotated dataset for a specific heritage site. The annotated images would be the source of annotations after the retrieval phase.
- Chapter 3 explains few basic optimizations that help in successfully developing an instance retrieval pipeline to run on a mobile phone.
- Chapter 4 makes use of visual phrases to solve the instance retrieval solution on an offline device within a small memory footprint.
- Chapter 5 concludes the thesis with finishing remarks and discusses the wide scope of other similar computer vision applications that could be designed for offline mobile phones.

Chapter 2

Efficient and Rich Annotations for Large Photo Collections

Large unstructured photo collections from Internet usually have distinguishable keyword tagging associated with the images. Photos collected from tourist and heritage sites of the world represent much more information than just a tag. The fascinating story of a historic monument, building or an architectural structure captured, is concealed in its photo. An art enthusiast may convince you that the art captures the soul of the artist or the architect. From our point of view, intriguing stories associated with historic structures are nothing but detailed annotations corresponding to its captured photos. Such photos can be described with detailed and part-wise annotations resulting in an improved automatic search and enhanced photo browsing experience.

Manually annotating a large community photo collection is a costly and redundant process as similar images share the same annotations. We demonstrate an interactive web-based annotation tool that allows multiple users to add, view, edit and suggest rich annotations for images in community photo collections. Since, distinct annotations could be few, we have an easy and efficient batch annotation approach using an image similarity graph, pre-computed with instance retrieval and matching for a given collection. This helps in seamlessly propagating annotations of the same objects or similar images across the entire dataset. We use a database of $20K$ images (Heritage-20K) taken from a world-famous heritage site to demonstrate and evaluate our annotation approach.

2.1 Introduction

The widespread usage of digital photography along with the Internet boom has made available billions of photographs over the Internet, particularly on social networking and photo sharing platforms like facebook, flickr, picasa, etc. The computer vision research community is busy in exploiting these large unstructured collections. Of particular interest are those "tagged" with famous tourist attractions. Community photo collections from the Internet have been used for creating interactive 3D photo browsing tools [46], reconstructing dense 3D scenes using multi-view stereo [15], summarising scenes [43], segmenting scenes [42], hole-filling [17], learning object category models [12], estimating geo-location [18] and finding image paths [45]. Most of these efforts take advantages of the quantity,

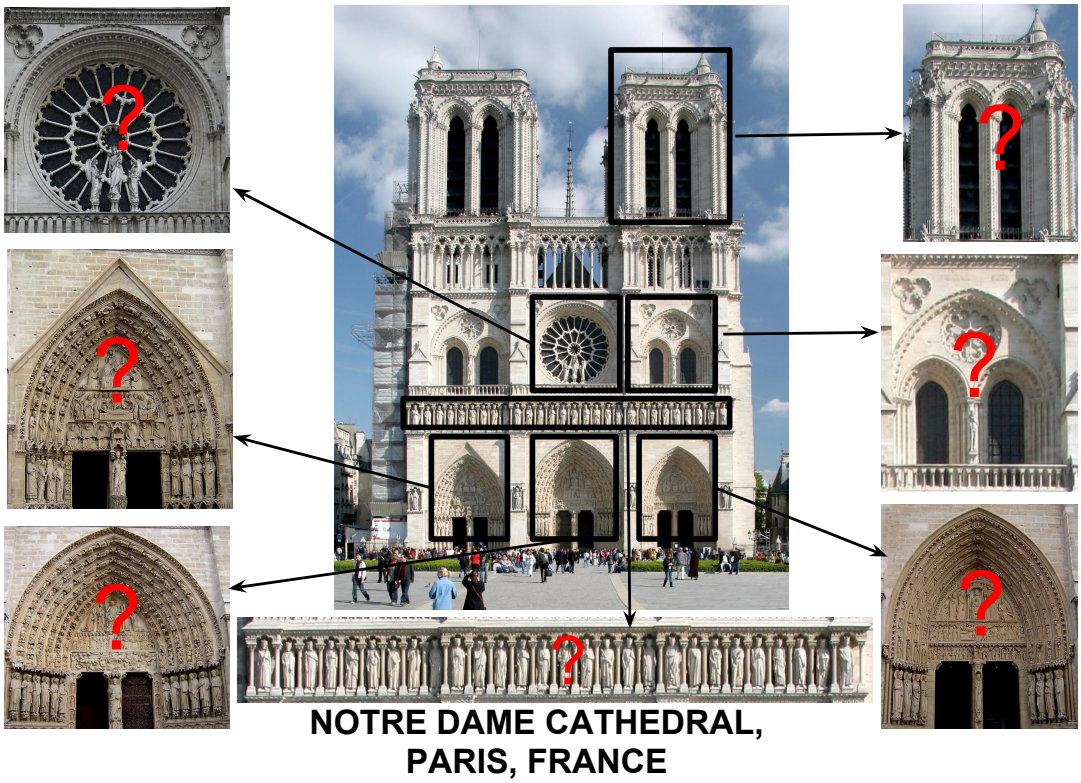


Figure 2.1: An Internet photograph depicting a popular monument, identified by its tag as "Notre Dame Cathedral" in "Paris, France" We are interested in particular details; for example, information about its history, architecture, etc as well as individual descriptions of highlighted objects that are usually interesting to an enthusiastic tourist.

variety and distribution of photos in a collection. Another important dimension is the similarity of the objects and scenes within a photo collection.

Internet photos are tagged by certain distinguishing keywords, which helps their retrieval by search engines. A large percentage of community photos have interesting scenes and objects within the image which could be associated with more informative tags than just vague keywords. Many such photos may come from popular tourist and heritage sites around the world. With the help of detailed tags or, an informative annotation associated with each image, a better structure is induced into the unorganised photo collection. This helps the cause of search engines as they have a larger text description for an image to look up the query keywords. Also, these self-explaining photos now enhance the photo browsing experience of a user interested in the collection. Another use-case for a richly annotated photo collection could be instant on-the-spot auto-annotation applications such as [35], that perform on-device instance recognition for photo captured using the camera of a mobile phone. These applications aim to educate the user about the scene or object captured in the query image. When the training dataset is well annotated, important details associated with the matched image can be presented to describe and

characterize the query. This has a great application in tourism as the mobile visual recognition app serves the purpose of an intuitive, automatic and consistent tour-guide for a particular location.

An image from a tourist site may pose several questions about the scene captured, the objects contained, the location specifics, the historical and/or archaeological significance, the architectural details, etc. All such necessary information can be associated with the images in the form of textual or audio/video annotations. Detailed annotations may also have separate parts or regions in the image marked to depict distinctive objects.

There are two major challenges associated with annotating a large collection of Internet images. Firstly, rich and precise information about objects in a photo is difficult to collect manually. Secondly, Internet photo collections may have sizes ranging millions of images. Annotating each photo one-by-one is a cumbersome task. However, the number of distinct scenes, buildings, or objects in the photos may be far lesser than the total collection i.e. many similar images may share the same annotations. Hence, an efficient mechanism to annotate a group of similar images at the same time could reduce the task sizeably.

We demonstrate a web-based community tool that helps collaborating with multiple users, who may be historians, students, tour guides or other people willing to share their knowledge. Once a photo is annotated, the tool tries to identify all photos in the collection which are similar to this or, have the same object/scene and thus, propagate the annotation across the entire dataset.

2.2 Image Annotations

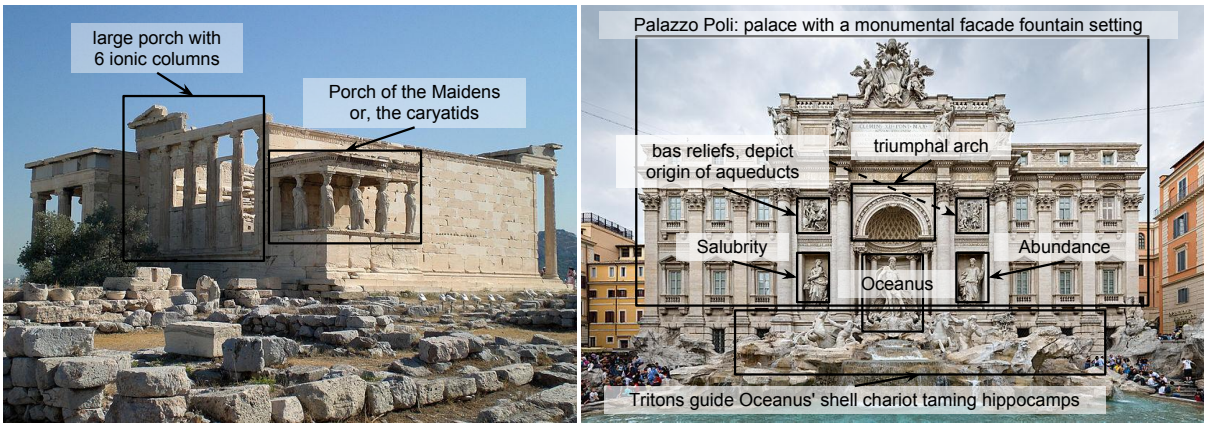
An image can be described with respect to the scene captured or the objects and structures present in it. Apart from the overall description of the image, one may be interested in specific objects occurring in certain parts of the scene captured. All such objects also need to be marked and annotated. Some objects may be partially occluded, which poses a difficulty in the annotation process. Mapping the annotation information with respect to each photo in the database, must be stored in a flexible format, so as to be easily integrated with various applications. In this section, we discuss how to richly annotate a photo.

2.2.1 Annotation Modes

We identify various types of annotations that could be added to describe a photo captured from a tourist site: text, hyperlink, audio, video, localisation on a map of the tourist site, object boundaries and graphical illustrations in the form of arrows for direction purposes. While adding text annotations is a straightforward task, other form of annotations require a generalised framework for capturing and storing information. Hyperlinks can be stored in form of text. Audio and video information can be stored by mapping the name of the corresponding media files to the image. The scene depicted in the image can be localised on a map of a tourist site by identifying the position on the map.

Figure 2.2: Example of scene instance annotations for a selected set of popular monuments of the world. These annotations carry important information about the monument or the structure in terms of its historic or architectural significance.

Figure 2.3: Example of scene instance annotations from the Heritage-20K dataset.



Erechtheion, Acropolis of Athens: Ancient Greek Temple in north side dedicated to Athena and Poseidon

Trevi Fountain, the largest Baroque fountain in the city; marks the terminal point of Aqua Virgo, that supplied water to ancient Rome

Figure 2.4: Example of object instance annotations demonstrated for a couple of popular monuments. Note the distinct part-wise annotation in the image in the form of marked rectangular boundaries around a specific object.

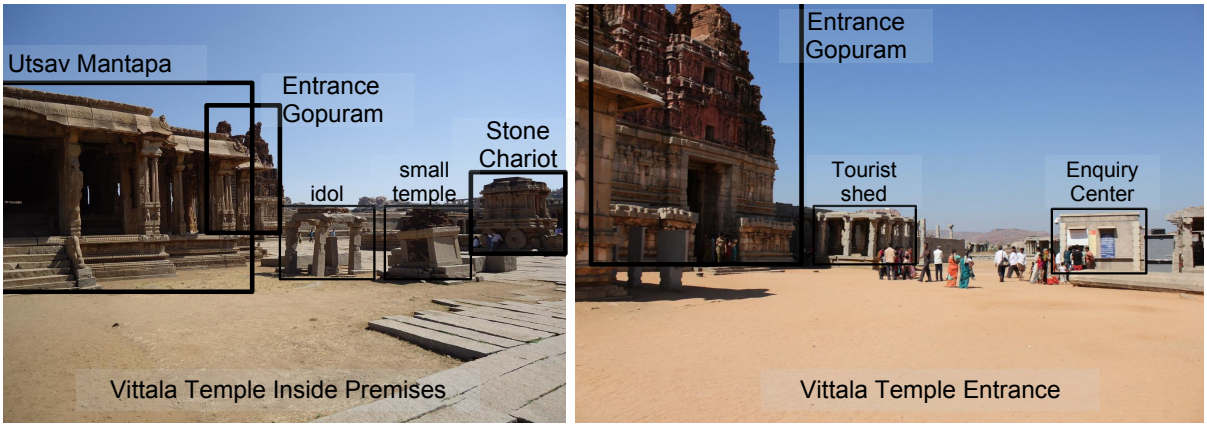


Figure 2.5: Example of object instance annotations demonstrated for a couple of structures from the Heritage-20K dataset. As in the above figure, object boundaries are represented with a rectangular border around them.

2.2.2 Scene and Object Instances

Scene instances are distinguished structures at a site which are of popular tourist and heritage interests. Annotations used to briefly describe the image scene may be called scene annotations (See Figure 2.2, 2.3). The description may relate to when a structure was built, who built it, what was it used for, and what historical or architectural significance it represents. It may also present some specific information related to various structures or objects present in the photo, or give a brief archaelogical detail. However, when multiple distinct objects are captured in a single image, a single overall description may not always fit.

Specific objects like an interesting artifact or an architectural structure occurring in a scene might have some interesting significance. It is very useful to identify and localise such distinct objects within the image (See Figure 2.4, 2.5). A rectangular boundary may be used to represent the object regions, along with storing the corresponding annotations. A photo may have multiple separately identifiable objects, in which case, it would require multiple rectangular bounding boxes annotating multiple objects. However, this presents a challenge in having an easy interface to mark rectangular boundaries on the image. Object instance annotations are different from the scene instance ones, particularly when a user finds only a particular region in the image depicting a distinct structure, and its description does not fit the entire photo.

2.2.3 Collaborating for rich annotations

Multiple users like historians, researchers, students, tour guides or other third parties may form a collaborative knowledge-based network to build strong useful annotations. While some collaborators would have in-depth knowledge of the history and significance of the heritage site, we could also get useful annotations from enthusiastic tourists who wish to share their experience.

The annotation propagating tool can be linked to social networking sites like facebook, flickr, etc where users bring in large number of personal photo collections from a particular site to share with their friends and family. They can contribute to our annotation database for the same heritage site by sharing their photos and captions with our tool. The annotation system is designed to work in a client-server fashion, which allows adding, modifying and deleting annotations by various users. This brings up a new challenge: Out of a large dataset of images, where should a user start annotating? The system must have an easy image browsing interface and highlight the "popular" images to annotate. With an easy and simple web-based platform, many enthusiastic tourists may come forward to share their travel experiences, and in turn, strengthen our annotation database. This is a form of crowd-sourcing technique, where information for specific monuments are obtained from various users and stored in the form of detailed and part-wise image annotations.

Further, with the availability of resources such as the *Knowledge Bank* for Hampi [25], we could easily make use of their data, to get better and detailed information about the stories associated with a

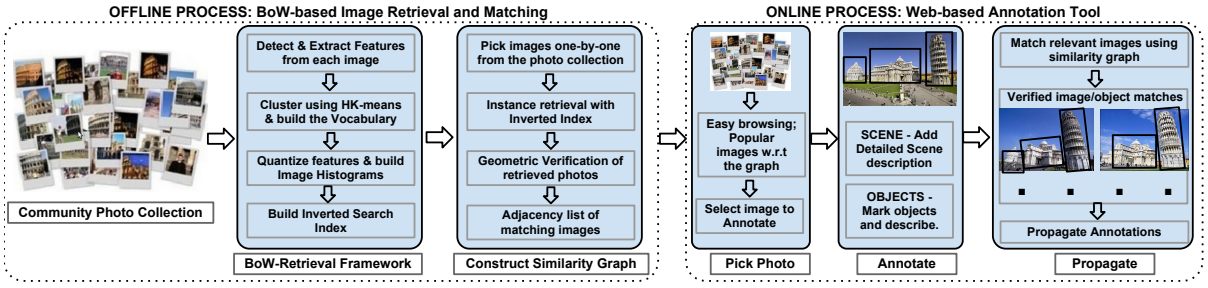


Figure 2.6: An overview of the annotation building framework. The offline process details the pipeline to construct an image similarity graph for a large photo collection. During the online process, we have a web-based annotation tool that allows easy browsing of the photos and selecting one to associate useful scene and object information. The annotations are then propagated across the collection.

particular tourist or heritage site. Working towards a common goal, this not only improves the prospects of our tool, but also helps the collaborating organization as well.

2.2.4 Representation

Annotations are mapped to the corresponding images and stored in an online database on the server. The database engine used for this purpose is mysql. Since, distinct annotations may be few, inorder to avoid anomalies in updations and deletions, we use a relative database model with 3-NF normalization. The database can be seamlessly exported into various formats like CSV, XML, JSON, YAML, etc so as per the requirement of the respective applications.

2.3 Efficient Batch Annotation Approach

In section 2.2, we present the annotation structure that helps associating each image with rich useful information. As discussed earlier, annotating each image in large community photo collections is a manually costly operation. The idea is to have an efficient and seamless annotation propagation approach to annotate a group of similar images simultaneously. Similar photos or same objects and structures occurring in many images share the same annotations. Once a photo is annotated, we look for matching images and object regions and spread the annotation across the entire collection.

2.3.1 Image Similarity Graph

Image matching across datasets having millions of images is a highly expensive method as it involves an exhaustive pairwise matching step. However, during the online query process, pairwise matching of the query image with only a small subset of the entire dataset is necessary. This subset must contain all

images from the dataset, that are similar to the query. An image similarity graph, mapping similarity and matching relations between photos in the collection, can be built to efficiently look-up large databases for a given annotated query photo. Each image in the similarity graph is a node and the edge between two similar images is weighted by the number of matching features. Further, the edges also store the pair-wise matches to speed-up the computations during online querying and matching the object regions. The resultant graph is a set of disjoint clusters of interconnected images and each cluster can be observed to contain spatially close images.

Constructing the graph involves an offline step of computing a set of similar images for each photo in the collection. This problem is related to the image retrieval problem, where a query image is given and the system returns a ranked list of similar images. We use the state-of-the-art instance retrieval approach (See section 2.3.2) for this purpose. In the graph-based representation of a photo collection, each image has an adjacency list of other similar images from the collection. This allows the discovery of neighboring images from the same scene, that may be visually distant from the query but share the same scene-instance annotation (See Section 2.3.3). Given a query, the pre-computed and verified list of images is then used to find corresponding matches for the object regions specified in the query annotation using object-boundary correspondence (See Section 2.3.4).

2.3.2 BoW Instance Retrieval for Similarity Relations

The similarity relations for each photo in a large dataset can be obtained by employing the Bag-of-Words (BoW) based instance retrieval approach [44, 38] to search similar images and objects with respect to a query image. This is followed by SIFT-based geometric verification of the retrieved images. The entire process, described in the following paragraphs, is applied to each image in the dataset to construct the Image Similarity Graph.

Given a set of images, local image feature descriptors like SIFT, SURF, etc are extracted at interest points of each image. These high-dimensional feature vectors are clustered using a clustering technique like K-Means to define a visual vocabulary for the image dataset. The features from each image are then, quantized and a histogram of visual word representation (also called BoW representation) is obtained for every image in the database. To speed-up the query process, an inverted search index is built, which maps the visual words to a posting list of images that contain them. During the retrieval process, the BoW representation is computed for the query image. For each visual word occurring in the query, a vote is given to all images that are mapped to it in the search index. A popular voting measure is the Term Frequency - Inverse Document Frequency (TF-IDF) based weighting of visual words occurring in relevant images. The database images are shortlisted and ranked based on these TF-IDF scores.

Once the similar images are efficiently retrieved, a filtering technique is needed to identify the false retrievals. For this purpose, geometric verification is used to remove errors due to outliers from mismatched or missing features, because of detector failure, occlusion, etc. The standard solution is to use the RANSAC algorithm [13], which involves generating affine transformation hypotheses using a minimal number of correspondences and then evaluating each hypothesis based on the number of "inliers"

among all features. This helps in estimating a fundamental matrix fit between the query and each of the target retrievals. The verified retrievals are scored corresponding to the number the inlier matches. These pair-wise verified matches are preserved in the edge information of the similarity graph. This later step typically improves the mean Average Precision (mAP) of the retrieval process i.e. we get a higher precision-recall values corresponding to the retrieval results.

2.3.3 Neighborhood relations in Similarity Graph

Figure 2.7: Identifying the same scene in two visually dissimilar images by exploiting the neighborhood relations in the Image Similarity Graph. As can be seen in the figure, the intermediary match helps to identify the similarity between the annotation source (blue outline) and the annotation destination (red outline).

As shown in Figure 2.7, an image having the same scene annotation as another could be visually dissimilar to each other. This occurs frequently in case of stereo images of a building or architectural structure with significant viewpoint variations at a heritage site. However, when we have a dense image dataset, it is possible to have intermediary matching images to suggest that two visually dissimilar images may actually come from the same scene. That is, an image I_3 may not be a direct neighbor of image I_1. However, if image I_2 is a neighbor of I_1, and image I_3 is a neighbor of I_2, then we can verify by the intersection of matching features, whether I_1 and I_3 come from the same scene. Thus, the image similarity graph helps verifying the neighbors-of-neighbors relations with strong edge-weights to identify images from the same scene-instance. This helps in propagating the scene annotations to a larger number of images.

2.3.4 Object-Boundary Correspondence

The objects annotated in the source query I_Q need to be automatically identified in all the retrieved and verified target images $I_1, I_2, ..I_N$. The list of matching feature keypoints between the source and the target, is obtained from the edge information in the similarity graph. These matches are already

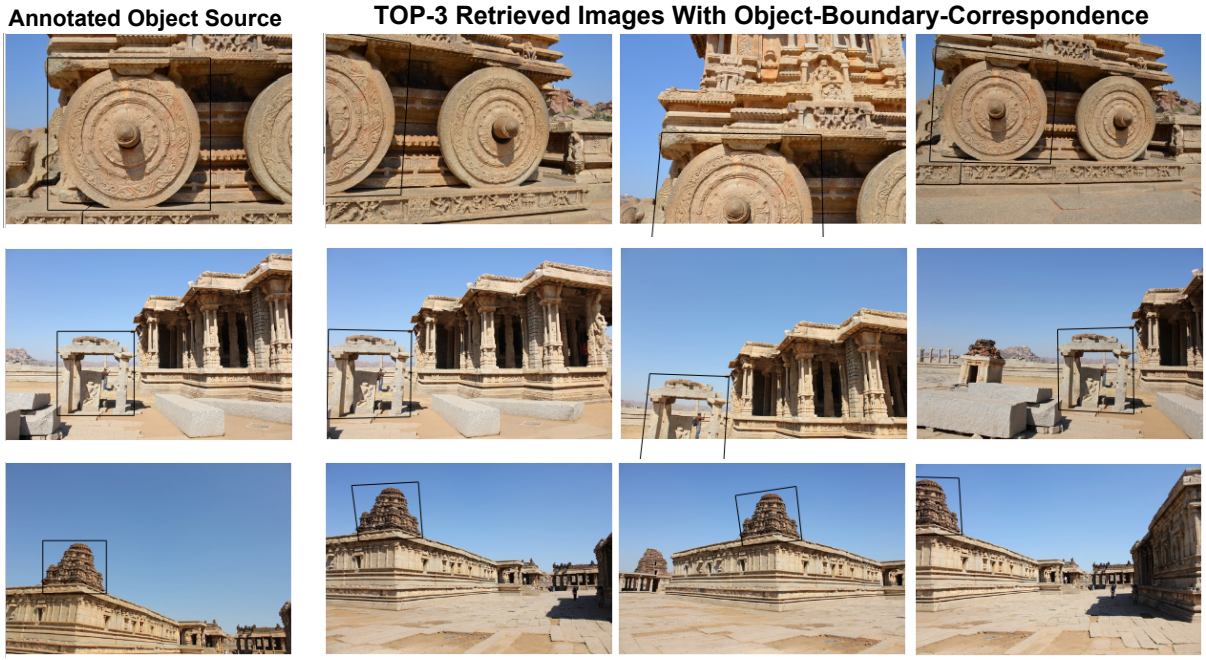


Figure 2.8: Illustration of Object-Boundary correspondence for an annotated object query source image and the boundary propagation shown for the top-3 retrieved results.

verified at the graph building stage. We now compute the homographies H_i's between I_Q and I_i, where $i = 1, 2, ..N$. Using perspective transformation, we can now project any point from I_Q onto the target images I_i's. The rectangular object boundary coordinates in the source image I_Q, localising the corresponding annotated objects, must be computed for each of the target images I_i(See Figure 2.8). For this, only four the corner points of the quadrilateral boundary need to be estimated in a target image to get the object-boundary correspondence. Now, the retrieved images also have rectangular boundaries over matching object regions. This can be again manually checked before propagating the object annotations to the target images.

2.4 Results and Discussions

The web-based annotation tool is built using *HTML5* and advanced *javascript* technologies. The server end uses a *mysql* database to centrally store annotations by different authors (collaborative users). The image browsing interface has been developed keeping in view the large size of databases that could be required to handle. We demonstrate the usage of the tool on a $20K$ image dataset from a famous heritage site. The image similarity graph for the dataset is pre-computed and stored in the form of *json* object files.

2.4.1 Heritage-20K dataset

Figure 2.9: Random sampling from the Heritage-20K dataset.

We introduce a novel data set of images from a world heritage site. This dataset has 20,958 medium-resolution (480 $\times$ 360) images of specific buildings, architectural details, interiors and exteriors of heritage monuments, etc. A random sampling from the dataset is shown in Figure 2.9. The challenge in annotating this dataset lies in connecting to the right information sources and collecting the tiny details about each architectural masterpiece. There are many similar looking beautiful structures (mostly stone carvings on pillars, walls, etc) with different significance and an enthusiastic visitor expects to learn most of it during his tour of the place.

2.4.2 Annotation Transfer using Similarity Graph

We use SIFT as the local image descriptors to extract features from each image. These 128-dimensional feature vectors are then quantized using Hierarchical K-Means to get a vocabulary tree, whose leaves represent the visual words i.e. cluster centers. The inverted search index is constructed as discussed in Section 2.3.2. Each image from the photo collection is then queried one-by-one and after the retrieved list of images are verified by geometric verification, the set of similar images along with their verified matching features are stored in an adjacency list. In this way, the Image Similarity Graph is constructed. Given a query image with annotated object regions, we use the pre-computed pairwise matches with the similar images in the similarity graph to compute a homography matrix between the query and each of the target photo. A perspective transformation is then used to map the object boundary coordinates from the query to the matched image. Further, the neighbor-of-neighbors relations in the similarity graph are exploited to identify near-scene images that share the same scene-instance annotations.

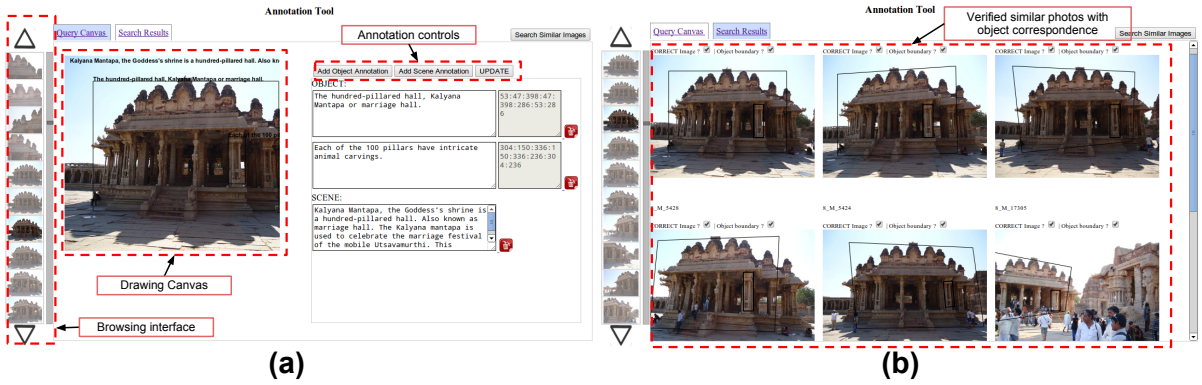


Figure 2.10: Snapshots of the web-based Annotation Tool allowing a user to select, annotate and propagate a suitable description for an image or an object. (a) displays the interface for browsing photos, the controls for adding annotations to a selected photo and the canvas where the image and object boundaries are drawn; (b) shows the list of similar photos, with corresponding object boundary regions, to which the source annotations can be propagated.

2.4.3 Annotation Tool

The tool has a simple interface to browse the thumbnails of the images from the dataset. The large number of image thumbnails are not downloaded at once at client side, but instead, fetched on-the-fly using *ajax* from the server, when the user scrolls through the collection. This makes the browsing experience fast, smooth and hassle-free. The already annotated images are shown within a green border on the thumbnail to distinguish from the non-annotated images. Also, listed are few popular images that are not yet associated with annotations. A popular image is chosen from the graph, if it is a part of a larger set of closely related similar images. Such photos may represent scenes or objects that are of interest to many. On selecting an image thumbnail, the full image appears on a *HTML5 canvas* and we have the controls for adding, modifying and deleting annotations for this image (See Figure 2.10-(a)).

There are separate controls for scene and object annotations. To add an object annotation, a boundary must be drawn over the image region on the canvas. This has been made simple with just two mouse clicks to represent the opposite corner vertices of a rectangular boundary. Once the annotations are added, the user can click and search for images from the database that are similar to the annotated photo. A list of images with corresponding object boundaries are displayed to the user. One can manually identify the false positives before clicking a button to propagate the annotations (See Figure 2.10-(b)). Each verified image is annotated in the database with the scene and object instances annotated in the source photo.

2.4.4 Evaluation

The effectiveness of the web-based tool built for the Heritage-20K photo collection, is measured in terms of the amount of manual efforts required to give rich and precise annotations to a set of images. A set of 30 distinct "popular" images were chosen for this purpose. For test purpose, only text annotations were considered for scene and object instances. The length of scene-instance text description for each of the 30 images was on average, 25-words or, 170-characters. Each image had 2 distinct object-instance annotations on an average, which implied that 2×2 clicks were required to mark the rectangular object boundaries. Each object was associated with a 10-word or, 70-characters description on an average. So, each image required typing 310 characters and 4 mouse clicks to associate a meaningful annotation.

Using our efficient batch annotation approach, each photo retrieved 20 verified similar photos on average and with a single click, the annotations for one image was propagated to 20 more images in a few seconds. So, at the end of the task, a total of 600 images were annotated. Thus, with the same efforts for annotating 30 photos, plus 30 extra clicks and a couple of minutes extra time, we annotated 600 photos instead of 30. This process was 20 times efficient than the naive way of annotating images one-by-one.

The task was assigned to 5 different users, who had visited the site before and had access to online and textual resources. Each user was assigned 6 photos. The user who completed the task earliest took 15 minutes and the most time taken by any user was 20 minutes. Hence, we were able to richly annotate 600 images in a short span of 20 minutes, which would have otherwise consumed days of a typical data-entry professional.

2.5 Conclusion

In this paper, we demonstrate a simple web-based scalable tool for annotating large image datasets, emphasizing the importance of rich content to be associated with every image. Target datasets are primarily, the community photo collections of tourist and heritage sites, where a lot of information with tiny interesting details could be built up for each distinct artifact or structure present. We design an easily navigable interface suitable for various types of users - historians, students, researchers, tour guides as well as casual tourists. The tool is developed to work in a client-server fashion. For easily and efficiently annotating a large number of similar images, we make use of the image similarity graph, which is pre-computed from the dataset. Bag-of-Words (BoW) based image retrieval and SIFT-based matching and verification is used to build the similarity graph. We demonstrate the usage of the tool to annotate a large photo collection of $20K$ images from a heritage site.

Chapter 3

Instance Retrieval on Mobiles

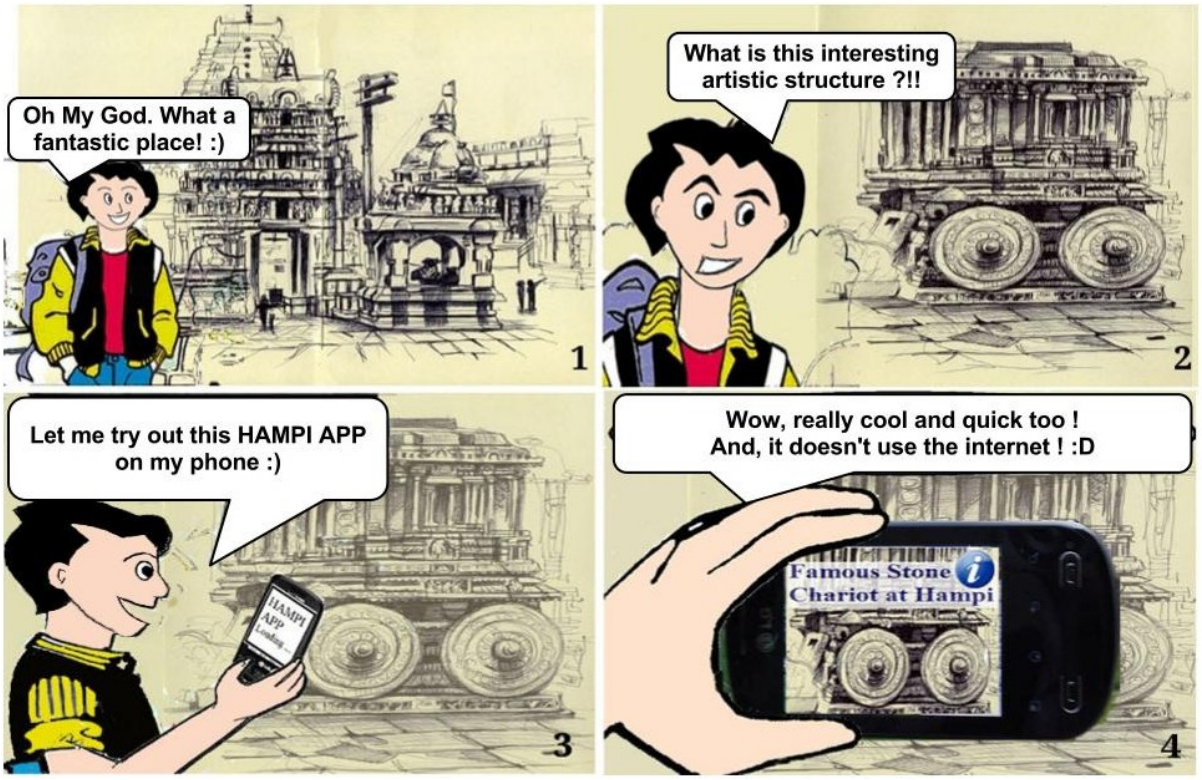


Figure 3.1: A typical use case for our mobile app, displaying auto-annotation at a temple in Hampi.

This chapter discusses the issues and challenges associated with developing a mobile app, where one can take a picture at a heritage site/monument and obtain associated annotations on a mid-end mobile phone instantly. In the previous chapter, we demonstrate how a curated dataset can be efficiently prepared for the use of such applications. Further, our application does not require any communication of images or features with a remote server, and all the necessary computations take place on the phone itself. We use simple techniques to build the solution and demonstrate the app on two Indian heritage

sites: Golkonda Fort and Hampi Temples. Underlying our application, we have a Bag of visual Words (BoW) image retrieval system, and an annotated database of images.

In the process of developing this mobile app, we extend the performance, scope and applicability of computer vision techniques: (i) we do a BoW-based image retrieval on mobile phones from a database of 10K images within 50 MB of storage and 10 MB of RAM. (ii) we introduce a vocabulary pruning method for reducing the vocabulary size. (iii) we design a simple method of database pruning, that helps in reducing the size of the inverted index by removing semantically similar images. In (ii) and (iii), we demonstrate how memory(RAM) and computational speed can be optimized without any loss in performance.

3.1 Introduction & Related Work

Existing mobile image instance retrieval applications assume a network-based usage where image features are sent to a server to query an online visual database. In this scenario, there are no restrictions on the size of the visual database. A wide spectrum of computer vision applications have been developed or delivered through mobile phones in recent years. These include commercial systems as well as research prototypes in the field of location recognition [41], product search [20], tourism [14], business [24], entertainment [16] and other generic image recognition apps like Amazon Snaptell [20], Nokia point and find [22], Google Goggles [23], Mobile Visual Search [3]. Most of these applications make use of the mobile Internet services to get a vision task solved on a remote server. Such applications, popularly known as apps, lie at the mercy of the mobile networks. They need considerable bandwidth to communicate with the server and are bound to have network delays. In addition, such services could also be expensive, if no other revenue is possible for the service provider. In the other approach, applications capture and process images realtime, in a standalone fashion with the limited storage and RAM constraints on the mobile handheld device.

Our work falls into the second category. We examine how to perform the task on an offline mobile device, where the entire visual index must reside on the mobile device itself within a small memory footprint. Such solutions have applications on location recognition and product recognition. We have the following use case. A tourist or a student, visits a heritage monument or site and is interested in specific artistic details of the structures. He/she queries the app with concerned images and the relevant information is returned as text (or even audio). There is no communication overhead or cost associated with such a *search*. Figure 3.1 shows the use case of our mobile app.

Technically we ask the following questions. Can we search and match instantly in a database of 10K medium resolution (480×320) images for relevant information on mid-end mobile phones which may have 600 MHz processor, less than 512 MB of RAM memory, and limited storage of 1 GB (including the SD-Card and internal memory). Can all the necessary computations be done on the phone itself so that the relevant information is retrieved within a second or two. To positively answer the above

questions, we innovate the Bag of Visual Words (BoW) based image retrieval pipeline in many ways. We discuss them in detail in Section 3.

The contributions of this work can be summarized as: (i) We demonstrate the effective and efficient retrieval of images from a reasonably large database (of the order of 10K images) on a common mobile phone, such that the necessary data is stored on the phone itself. (ii) We design simple vocabulary pruning methods which result in reducing the RAM usage and improving the computation speed without affecting the performance of retrieval. In fact, we obtain better performance with a pruned vocabulary of 5K compared to a k-means clustered 5K vocabulary for the Oxford Buildings dataset. (iii) We prune our image database to reduce the Inverted Index storage on mobile phones, which further reduces the RAM usage. We do this by removing semantically similar images from the database. (iv) We develop applications for auto-annotation and image-based GPS (Pseudo-GPS) for tourist purposes on a typical heritage site. (v) We demonstrate a collaborative annotation scheme suited for extending to a community project. Deshpande *et al.* [11] provide an application for displaying user photo collections on a 3D point cloud, which could help in 3D visualization and annotating the images easily. (vi) We quantitatively and qualitatively verify the solution on two heritage data sets — Hampi and Golkonda. We test our app with over 2000 annotated images from these sites using queries captured with low and mid-end mobile phone cameras.

With the increasing pervasiveness of mobile handheld devices, the field of mobile visual search is largely gaining interests in the research community. In such mobile vision apps, an image is taken using the mobile embedded camera, which is further processed for matching against a reference database of a large number of images. Typically, these apps follow a client-server approach, where the image or its processed features are sent from the client mobile to a remote server for actual processing. In such a model, a lot of work has been done in extracting compact descriptors from the image captured by mobiles, in order to reduce the network latency in server communication [8, 30, 7, 5, 6, 31]. However, if the mobile network itself is weak for a stable internet data connection, the method fails. The other approach is where the client downloads the necessary data from the server beforehand, and all vision algorithms are run on the handheld device [14, 47, 50, 19]. With the necessary data loaded for on-device processing, it can run a standalone app. This approach overcomes the network bandwidth issues that posed a major bottleneck in the former approach, but is bound to use the limited computing resources on the mobile device. Henze *et al.* [19] and Fockler *et al.* [14] demonstrate such mobile apps for efficient object recognition and work with dataset sizes of the order of hundreds of images. In this paper, we demonstrate a mobile visual search app that works with thousands of images.

3.2 Instance Retrieval with BoW

We address the problem of Instance Retrieval, similar to the retrieval problem posed for the standard Oxford Buildings dataset [38]. The web-scale landmark recognition system [55] is a good practical application of instance retrieval.

Our solution uses image retrieval and matching as the basic modules. This is implemented in a bag of visual words (BoW) framework using the popular SIFT-BoW pipeline [44] for retrieval. Given a set of database images, SIFT vectors are extracted at interest points. A subset of these SIFT vectors are first clustered using K-Means to define a vocabulary. Then all the SIFT vectors are quantized and a histogram of visual word representation is obtained for every image in the database. All these images are then represented using an inverted index. Given a query image, one can retrieve the images which share enough visual words with the query. While comparing query and database images, their relevance based on an information criteria (TF-IDF) is used [44]. This retrieval can in fact be done very efficiently, say in sub seconds from millions of images on a typical desktop. However, the RAM and storage requirements are high for a mobile application.

The BoW model fails to incorporate the spatial information into the ranking of retrieved images. In order to confirm image similarity, the visual words in the retrieved image must be spatially consistent with those in the query image. Once the similar images are efficiently retrieved, they are verified, as to whether they are the same object or not, by fitting a fundamental matrix (See Figure 1.6) over the point correspondences between query and retrieved images. This typically improves the mean Average Precision (mAP) of the retrieval process in instance retrieval [38]. However, it involves matching the high dimensional descriptors in query image with those in the target image to obtain a minimal number of correspondences, which then helps in generating a transformation hypothesis.

After this, our task is to annotate the matched object in the query image. The images in our database are prior annotated according to the scene or object present as discussed in Chapter 2. On finding the best match to the query, our task is to transfer the corresponding annotation to the query image. The entire process when performed on a typical desktop, needs considerable amount of main memory(RAM), storage and processing power. Apart from storing the images, The RANSAC based spatial verification of two images uses the 128-dimensional SIFT vectors and this also needs to be stored.

In order to accomplish this on a typical mobile phone, we need to store data on the phone itself. Extracting SIFT descriptors and performing spatial verification, are two computationally expensive steps on a mobile processor. The practical bottlenecks are the storage and RAM requirements in the entire process. Let us look at the storage requirement for a typical image retrieval application in this setting.

1. 10K images of size 480×320 requires a typical storage of 1.5 GB even if stored in a compressed jpg format.

2. Such an image typically has around 500 interest points, each represented using 128-dimensional SIFT vectors and their keypoint locations. This comes to around 1.5 GB of storage.

3. Each image typically gets represented using a histogram of size 5K bytes. This leads to 50 MB.

4. Each annotation of around 100 B, is stored in text format for all the images in the database. This needs around 1MB of storage.

Our immediate challenge is to do this on a mobile phone with 800 MHz processor, using a maximum RAM of 15 MB and get results in close to a second. We bound our storage requirements to 60 MB, which can be available on the internal memory or the SD-card of the mobile phone.

3.3 Retrieval and Matching on Mobiles

Mobile devices are limited by their computation and storage capabilities. We target a specific heritage site and build a vocabulary specific to the images in and around this location. The number of images may be as large as 5K-100K for a typical site including images that capture the intricate architectural details as well. Accordingly, the vocabulary for BoW-based retrieval process may vary from 5K to 1M visual words. We demonstrate that our mobile app can efficiently retrieve annotations in such scenarios. In this section, we also experiment with the popular Oxford Buildings dataset [38] to demonstrate the advantages of the enhanced BoW.

3.3.1 Retrieval without Images

An image retrieval application gives a ranked list of images as its final output. However, our mobile app is concerned with the annotations on an image and not the image itself. Hence, we map the images to their corresponding annotation information and store only the necessary image features and data required for our mobile app. We do offline computations such as vocabulary building and inverted index creation on a server. This is a one-time effort. We use scalable vocabulary trees [34] for our vocabulary construction and store this on the mobile phone along with the inverted index. When the application starts on the device, these are read into the main memory(RAM), taking up less than 10 MB and this is sufficient for our BoW-based retrieval.

During the online retrieval stage, a user queries an image or video frame for annotations. The mobile device processes this image to retrieve a ranked list of k images based on their TF-IDF scores. These top k images are chosen for spatial re-ranking. However, we still face the challenge to store the high-dimensional SIFT vectors which are essential for spatial verification between the query and retrieved images.

3.3.2 Fast and Compact Spatial Re-ranking

Instead of using 128-dimensional SIFT descriptors, we simply use the quantized visual words corresponding to the keypoint locations to check whether the retrieved image is spatially consistent. We compare our results of visual words matching Vs SIFT matching (using the full-length descriptors) on the standard Oxford Dataset. While using the full-length descriptors gives a mean Average Precision(mAP) of 60.73%, using only the visual words, the mAP reduces to 57.55%. However, our application is concerned with only the top image, which acts as the annotation source. Hence, we re-rank only the top 5

images. So, we compute the precision at rank 5 using visual words matching for spatial re-ranking. It comes to 90%, which remains same even when the SIFT descriptors are used in matching.

With this, our storage and RAM requirements are lowered. We store the keypoints and their corresponding visual words that take up 36MB, preferably on the SD-card of the mobile phone. During spatial verification, the corresponding file for the retrieved image, of around 8 KB is read from the SD-card and copied into the RAM.

During spatial verification, a keypoint K_i with visual word as V_i in the query frame, matches with a keypoint K_j with V_j in the retrieved image if, both are represented by the same visual word i.e. if $V_i = V_j$. Therefore, instead of computing L2-distance and comparing for each pair of the 128-dimensional descriptors, we compare only two integers. This speeds up our spatial verification step.

3.3.3 Annotation Accuracy

Typical Image retrieval systems evaluate the performance using the average precision (AP) measure computed as the area under the precision-recall curve for a query image. For a set of test images, the mean Average Precision (mAP) evaluates the overall performance.

We are interested in Precision at Rank-1, since the top retrieved image is considered as the best match in the database and acts as our annotation source. Hence, we choose to evaluate the performance of our mobile app with a different measure. We use the Precision at Rank-1 to say whether a query image has been successfully annotated or not. We call this as the Annotation Accuracy(AA).

In order to compute Annotation Accuracy for our Heritage App, we collect test images of *N* buildings and structures using mobile phone cameras at a particular site. This is our test dataset. Annotation Accuracy is computed for each of the *N* monuments and averaged to give the mean Annotation Accuracy(mAA), which evaluates the performance of our Heritage App for a specific site. Our methods of optimization may not be applicable for a generic search.

3.3.4 Vocabulary Pruning

We target to run our application on low and mid-end mobile phones. On such devices with low processing capabilities, the percentage of RAM used during online query processing becomes an important factor. The vocabulary and the inverted index that we need for retrieval on the mobile phone, consume most of the RAM during the query processing stage. We reduce RAM usage by reducing the vocabulary size without affecting the retrieval performance. This, we call as vocabulary pruning.

Our approach in pruning the vocabulary is by removing the visual words that are less important. We do this in two different ways. Firstly, we intuitively decide the importance of visual words by analysing the number of images they appear in. Suppose, a visual word V_i is indexed to n_i images. We set a upper threshold τ_H and a lower threshold τ_L. If τ_L ¿= n_i or, if n_i ¿= τ_H, then we remove this visual word, arguing that it is less likely to be discriminating. We analyse the results of this approach on the standard

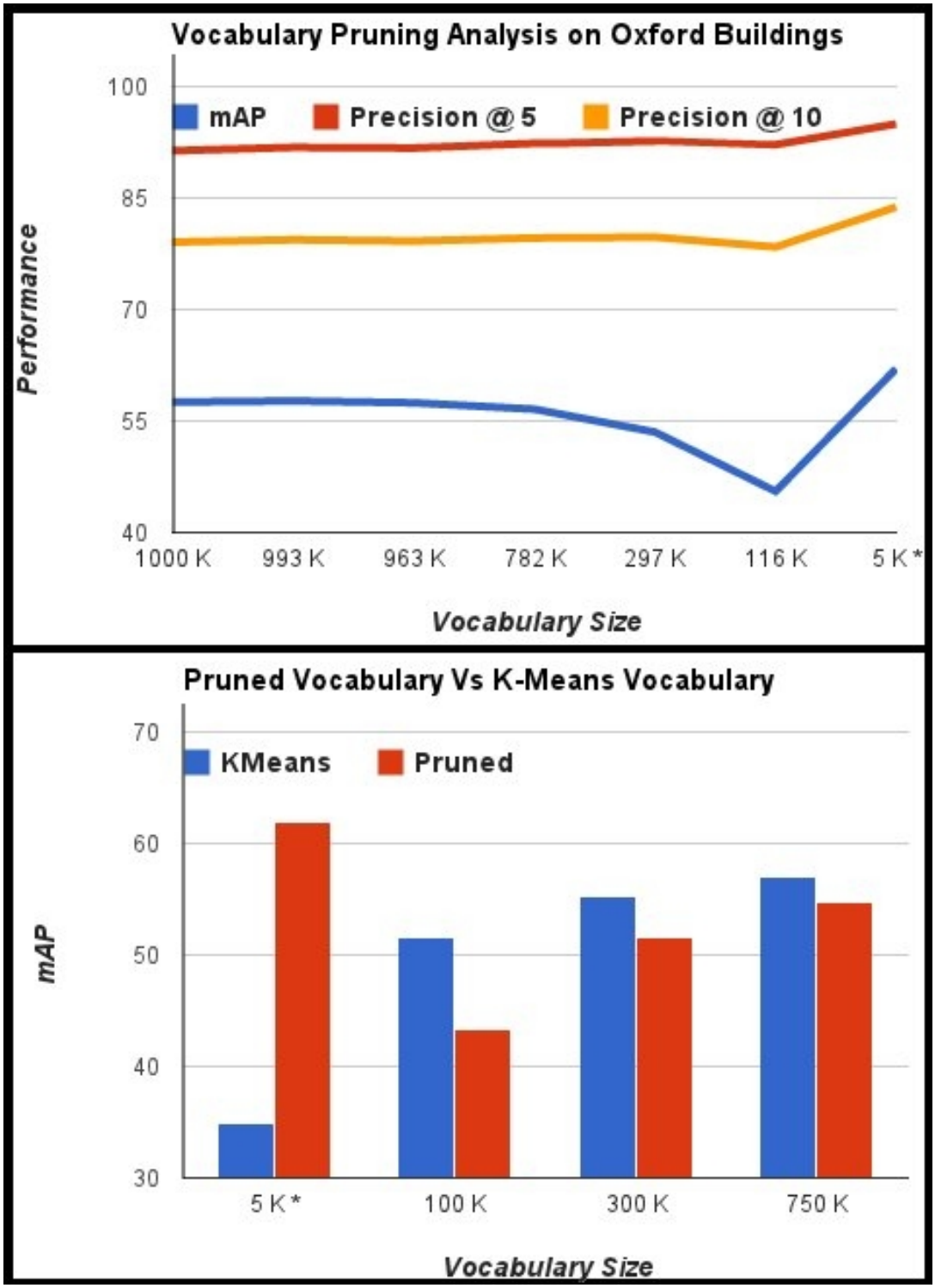


Figure 3.2: Performance analysis for different pruned vocabularies. *: In the supervised method, we get one pruned vocabulary of size 5K, that performs better than others. The other sizes ¿100K were obtained in the unsupervised method.

	Oxford Buildings	Golkonda
Total Images	5,062	5,500
Pruned Database	3,206	3,536
Original Inverted Index	99 MB	7.9 MB
New Inverted Index	76 MB	4.4 MB
mean AP (before)	57.55%	-
mean AP (after)	57.06%	-
Precision at 1(before)	92.73%	96%
Precision at 1(after)	97.27%	94%

Table 3.1: Database Pruning Results on Oxford Buildings and Golkonda datasets

Oxford Buildings dataset. It is observed that the mean Average Precision(mAP) reduces with the size of the vocabulary. However, the Precision-at-5 and Precision-at-10 remains unaffected(See Figure 3.2).

In another approach, we follow a supervised pruning technique. We use the ground truth images to identify those visual words that result in wrong retrievals. We start with a training set of labeled images. Initially, each visual word V_i is given zero score. We perform retrieval for each image in the training set. Let us consider the retrieval process for an image I_i. A visual word V_j occurring in the image gives TF-IDF scores to other database images, say J_k, in which it occurs. Now, suppose g_i : ground truth set for image I_i;
if $J_k \in g_i$, then V_j's score is incremented by the TF-IDF value, else its score is decremented.

Hence, after iterating through each I_i, every visual word V_i gets a final score S_i. We observed that, out of a total vocabulary of 1M for the Oxford Buildings dataset, (i) only 5K visual words have S_i ¿ 0, and hence cause a "positive" impact on the retrieval process. (ii) 76K visual words have S_i ¡ 0, and cause a "negative" impact. (iii) Rest 919K visual words have $S_i = 0$, and do not affect the retrieval process.

We use the "positive" 5K visual words for our retrieval. The mAP increases by 4% (See Figure 3.2 (a)) when evaluated using a different test dataset. Hence, with 5K visual words, the vocabulary is reduced by 200 times and yet we see a performance improvement. This approach is constrained to use an extensive ground truth, which is available with us in the form of an annotated database. We expect our app to work excellently on images where annotation is available. On images without annotations, we anyway cannot do anything.

3.3.5 Database Pruning

In section 3.3.4, we optimized memory usage on mobile phones by reducing the vocabulary size. The other dimension in the inverted index matrix is that of the number of images in the dataset. We usually have a lot of images in our dataset that are semantically similar to each other and hence, can be termed

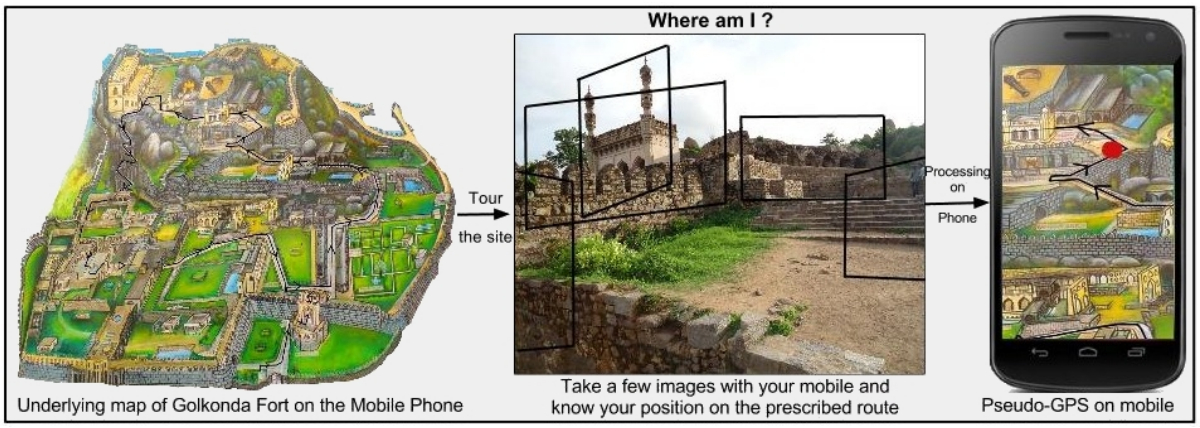


Figure 3.3: Usage of image-based Pseudo-GPS to find the current location on the prescribed route of Golkonda Fort

as repetitive. We remove these images from the inverted index without sacrificing on the performance. This is called Database Pruning.

We perform a Reverse Nearest Neighbors(RNN) search in the total set of images in our database. We exhaustively find the closest match X_j, for every image X_i. In order to find the closest match of an image X_i, we do a BoW-based retrieval followed by spatial re-ranking, and the top-ranked image X_j is considered the closest match. Now, X_j has X_i as one of its RNN. After doing this for all images in the database, we identify images that have zero RNN. Since, these zero-RNN images never occurred at top of the retrieval results, we remove their entries from the inverted index. This way, the database is pruned to reduce the size of the inverted index matrix, which helps in reducing the memory(RAM) consumption by our application.

We demonstrate results on the standard Oxford Buildings Dataset along with our Golkonda dataset(See Table 3.1). In both the datasets, we were able to reduce the database by around 35%, without affecting the performance. As observed in the table, the size of the Inverted Index reduces by 3.5 MB for the Golkonda Dataset, and by 21 MB in case of Oxford Buildings.

3.4 Applications in Digital Heritage

In this section, we demonstrate the applications of our Heritage App on mobile phones. We primarily focus on generating annotations for a query image to describe the object or scene present. We extend the app's capability to generate position-related information. We also contribute a prototype social tool, meant for setting up a platform for collaborative image-based annotations.

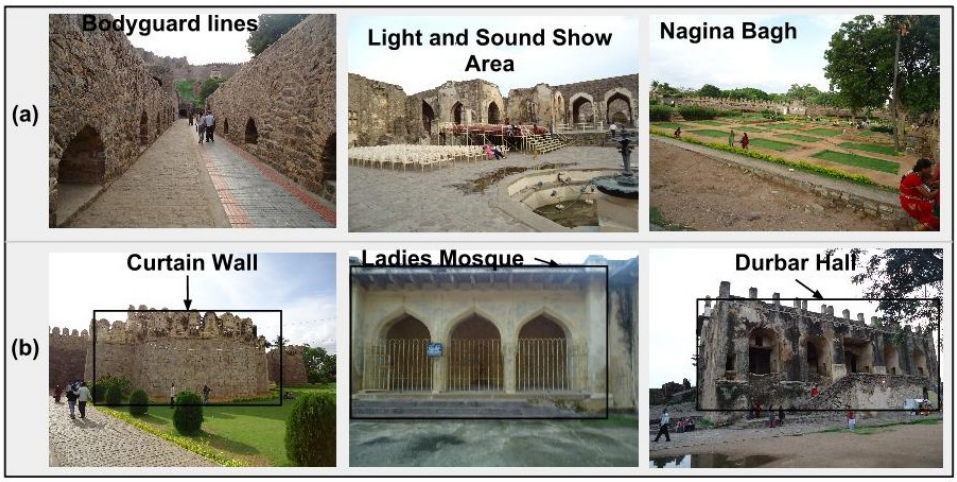


Figure 3.4: Sample images with (a)Scene and (b)Object annotations

Figure 3.5: Mobile Heritage App Illustration for 3 different scenes at Golkonda Fort

3.4.1 Scene and Object Annotations

We choose to classify annotations into two different categories: scene-based and object-based (See Figure 3.4). Scene annotations are like image captions where we store information related to the buildings or structures present.

Object annotations are where a particular structure, building, or an interesting architecture is marked by a rectangular boundary on the image and a description is stored for it. When we find such annotations in the annotation source (the best match for a query), we mark the object's boundary on the query image. The object boundary information is stored in the form of four corner points representing a rectangular region in the image. In order to transfer the object boundary from the source image, we estimate a homography with the query image. For this, we use the inlier matches computed during the spatial verification step. Based on the computed homography matrix, we perform a perspective transformation and get a point-correspondence for the four corner points, which represent boundary coordinates of the matched object in the query image.

3.4.2 Pseudo-GPS Navigation

A heritage or historic place is usually spread over a large geographic area with interesting structures at different locations within the site. There exists a specified route to be taken for some sites to easily tour the place.

We enable a tourist to look-up his current position on the map and guide himself on the prescribed route at a heritage site. On the mobile device, we store a map of the place, with the important structures, buildings and other objects and regions marked out clearly. A tourist can click a few images from his current location on the marked route and query our application for his position. The map also displays the other important structures or scenes near by and directs the tourist on the prescribed route for exploring the site. We call this functionality as Pseudo-GPS Navigation as it works similar to GPS systems, but only using images.

We demonstrate this application at Golkonda Fort(See Figure 3.3), where there is a prescribed route (approximately 2km long) to tour the site. We collected training images from 43 locations, separated by a few meters on the route. Each such location is a nodal point, which helps in identifying a tourist's position, discretely on the map. Ideally, each nodal point spans approximately 4-5 meters and is separated from its next nodal point by 10-11 meters. As a tourist tries to find his location on the route, he clicks 5-10 images of intuitively distinct structures visible from his current location. Our application performs BoW-based image retrieval for each of these query images and scores the nodal points according to the images retrieved. The location corresponding to the nodal point with highest score is highlighted as the current position of the tourist on the map. Here, the query images are annotated by a position index on the map.

3.4.3 Scalable Social Annotation Building

We also developed a working prototype of an Annotation Building System. This allows users to upload an image, annotate it and propagate the annotation in the existing database. It facilitates both scene and object annotations on images. With the help of this tool, we have annotated over 2000 images in our Golkonda and Hampi datasets. We propose to make this tool available to users through the World Wide Web and via the social networking platforms. People adding photos with captions on their facebook or flickr profiles can propagate it to our database.

In view of the scope of our mobile app, the initial dataset we start with, might miss out a few important buildings, structures, or their different views. Through this tool, we seek to incorporate new images and annotations in the future releases of the app.

3.5 Implementation and Results

Our aim is to power a mobile handheld device with an end-to-end application specific to a heritage site. This application makes use of the mobile camera to capture images or video frames of interesting

	Time(in seconds)
App Loading	
Reading Data	12 s
Frame Processing	
SIFT Detection	0.250 s
SIFT Descriptor Extraction	0.270 s
Assigning to Vocabulary	0.010 s
Inverted Index Search	0.260 s
Spatial Re-ranking	0.640 s
Annotation Retrieval	0.010 s
Total	1.440 s

Table 3.2: Time Analysis for the Heritage App on a mobile phone with 800 MHz processor and 512 MB RAM

monumental structures and retrieve related information. In order to evaluate the performance of our application, we look at two factors: (i) how accurately are we retrieving information. (ii) how quick is the retrieval for a query image on mobile. The first factor is evaluated by Annotation Accuracy as defined in section 3. The second factor is evaluated by the real run time of the application to process a query image/frame on a mobile phone while efficiently using the memory.

3.5.1 Dataset and Annotations

We demonstrate the methods discussed in this chapter on two popular tourist heritage destinations in India: Golconda Fort, Hyderabad and Hampi Temples, Karnataka. For both the sites, we start with a dataset of around 5K images covering most parts or locations at the site. Using this as the training set, a BoW-based image retrieval pipeline is implemented separately for each site. We collected annotations for our images by visiting these heritage sites and acquiring details on monuments, buildings and other interesting architectural structures. Then we propagated these annotations across the images in our dataset using the Annotation Building Tool (Section 3.4.3).

The necessary data for retrieval, that was computed offline, is transferred to the mobile phone during the installation of the app. The data corresponding to image-wise annotations is also stored in a compact index on the phone. An image is mapped to one or more distinct annotations. For the Golconda dataset, we have annotations for 45 distinct scenes and objects, in more than 1500 images. Similarly, for the Hampi dataset, we managed to collect 20 distinct scene and object annotations across 500 images.

Site	Monuments	Queries	Annotation Accuracy
Golconda	14	168	96%
Hampi	10	60	93%

Table 3.3: Performance Analysis: Mean Annotation Accuracy for Heritage App for Golkonda Fort and Hampi Temples

3.5.2 Android Application

We demonstrate our system on mobile handheld devices (phones and tablets) with Android operating system. We use OpenCV4Android library to perform the computer vision and image processing tasks on the device. We need the user to pre-install the site-specific application on his mobile device before visiting the place. Once the application is installed and the required data is stored, it is ready to be used at the site.

When the application starts, the vocabulary and inverted index is read from the phone's storage. A user can now click an image or select a video frame as a query. We use OpenCV's methods to detect and compute 128-bit SIFT descriptors for the query frame. These descriptors are then quantized into visual words using the vocabulary tree. The application then does a quick TF-IDF based scoring of all the images in the dataset using the inverted index and finds a list of top-5 best matches from the database. These images are spatially verified and re-ranked to identify the annotation source. The annotation text corresponding to this source is displayed on the query frame. If an object is marked, we also obtain its boundary coordinates on the query image by performing a perspective transformation with an estimated homography between the two images.

3.5.3 Results on Golkonda Fort

The Golkonda Fort complex is spread over 7 km, in circumference. Mostly in ruins, it still has many interesting buildings and structures. We worked with a dataset of 5,500 images spread over the entire fort. For testing our app, we identified 14 significant structural buildings and collected a set of 10-15 test queries at each of these scenes. These query images (See Figure 3.6) were taken using low-end mobile phones with resolution as low as 3 MP. We evaluate the application's performance on these queries and achieve 96% mean Annotation Accuracy (See Table 3.3). We also observe the time taken for processing a query image on the mobile device (See Table 3.2). The annotations for query frames are retrieved and displayed on the screen of the mobile device in an average time of 1.5 seconds. We analyse the feature data storage and RAM usage on the mobile phone for our app on Golkonda (See Table 3.4). Our app uses occupies ¡50 MB on the SD-card or internal memory storage of the mobile phone and uses ¡10 MB of RAM during runtime.

We also evaluate our Image-Based Pseudo-GPS application on the 2 km prescribed tourist route at Golkonda Fort. As discussed in section 4.2, we train our system to discretely identify the location of a

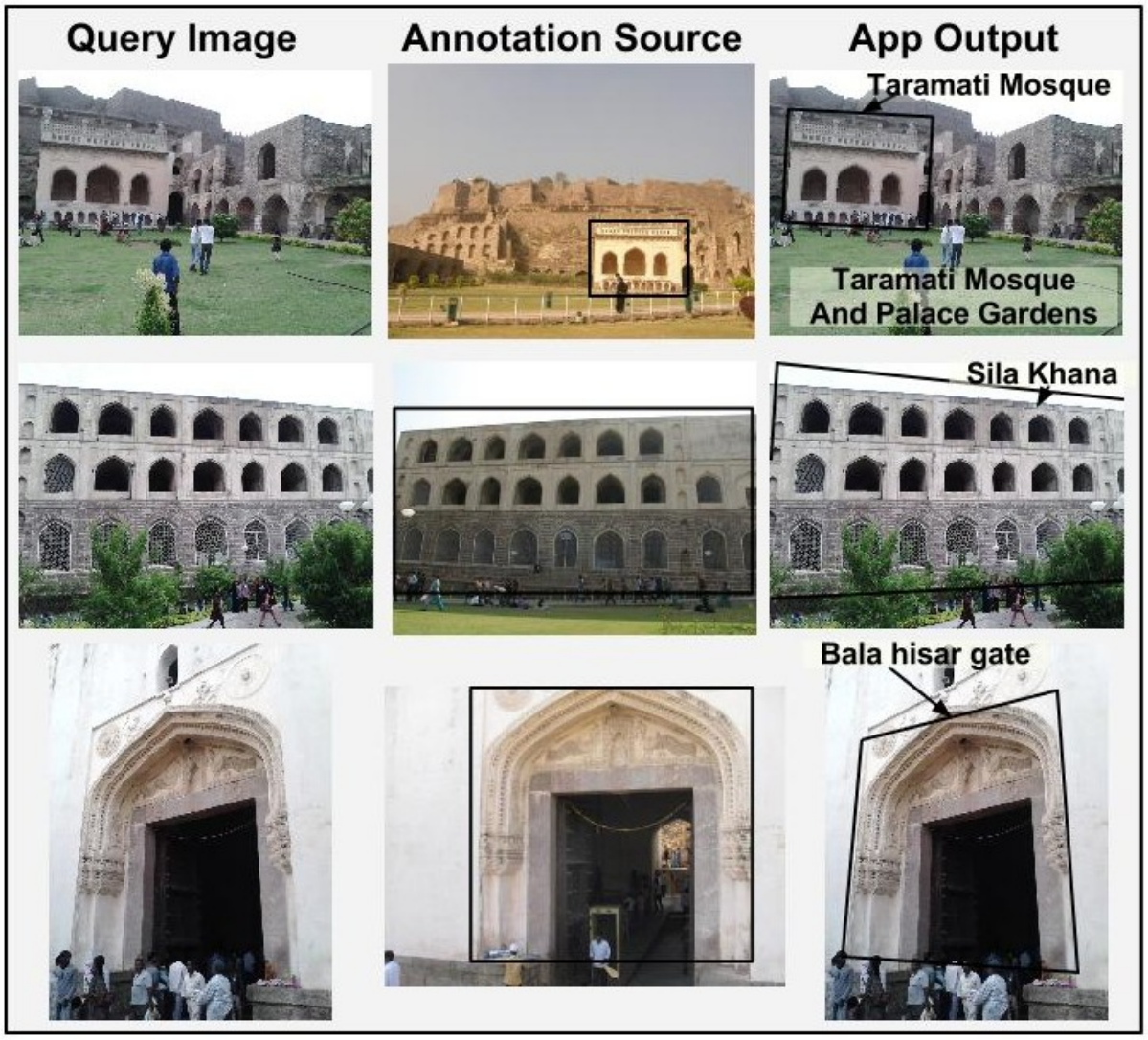


Figure 3.6: Annotation retrieval for three sample query images at Golkonda Fort.

tourist as one of the 43 nodal points on this route. We exhaustively test our application by using phone-captured images from each of the 43 locations. A query at each nodal point consists of 5-10 images that intuitively capture distinct structures, or near-by monuments visible from that location. We observe that 41 out of 43 locations are correctly recognized. The two failure cases, are classified into their nearest nodal points. This can be attributed to visually similar structures captured from near-by locations.

Our mobile app performs poorly if the query image has crowd presence (See Figure 3.7). Therefore, our app's utility is limited at crowded sites. Most of our query images, that we have tested, have limited presence of people and crowd.

	SD-Card	RAM
Vocabulary (10K Visual Words)	2.4 MB	2.4 MB
Inverted Index	4.4 MB	4.4 MB
Quantized Words and Keypoints	36 MB	8.0 KB
Annotations	88 KB	88 KB
Total	42.888 MB	6.896 MB

Table 3.4: Storage and RAM usage analysis for Heritage App on a Mobile phone for the 5K Golkonda dataset

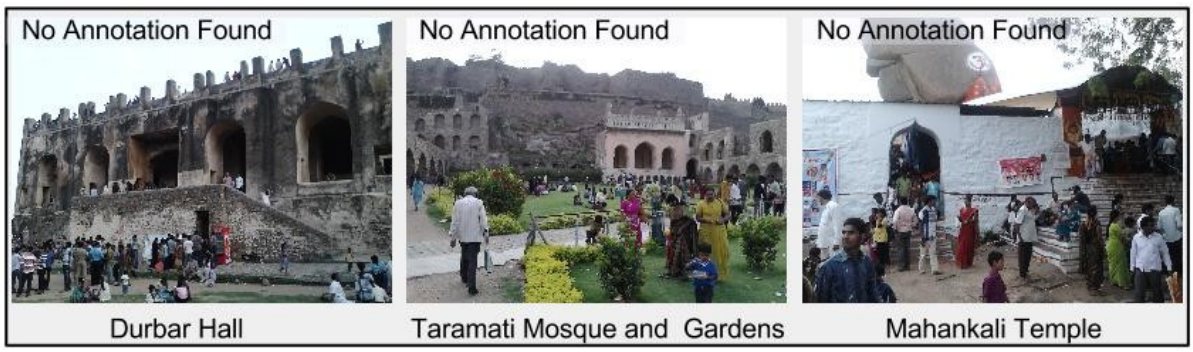


Figure 3.7: Query Images with crowd presence, for which our Heritage App fails in retrieving correct annotations from the database.

3.5.4 Results on Hampi Temples

Hampi Temples are spread over a wide region unlike the Golkonda Fort. We collected images from different temples and built a dataset of 5,718 images. This dataset is largely comprised of images covering temple overviews and structures and does not encompass much of the internal architecture, which would be a much denser dataset. Similar to experiments at Golkonda, we choose 10 important monuments at different temples of Hampi to evaluate the Annotation Accuracy on phone-captured query images. We achieve an overall 93% mean Annotation Accuracy.

In another experiment at the *Hazara Rama Temple, Hampi*, we collected an image dataset covering the internal architectural details. This includes images of numerous sculpted friezes depicting the ancient Indian epic, *Ramayana*. These relief structures, stone-carved on the walls of the temple particularly around the main shrine present the epic story in the form of distinctive scenes(See Figure 3.8(a)). We find that the previous vocabulary built on the 5.7K dataset is unsuitable for these images, as there are fewer keypoints owing to the distinct texture of the stone carvings. We compute a new vocabulary from these images and build a BoW retrieval system. Our application tries to identify and annotate each of these scenes, when captured using a mobile phone. In this way, a tourist can guide himself through the interesting mythological story of *Ramayana* at this 15th century shrine (See Figure 3.8(b)).

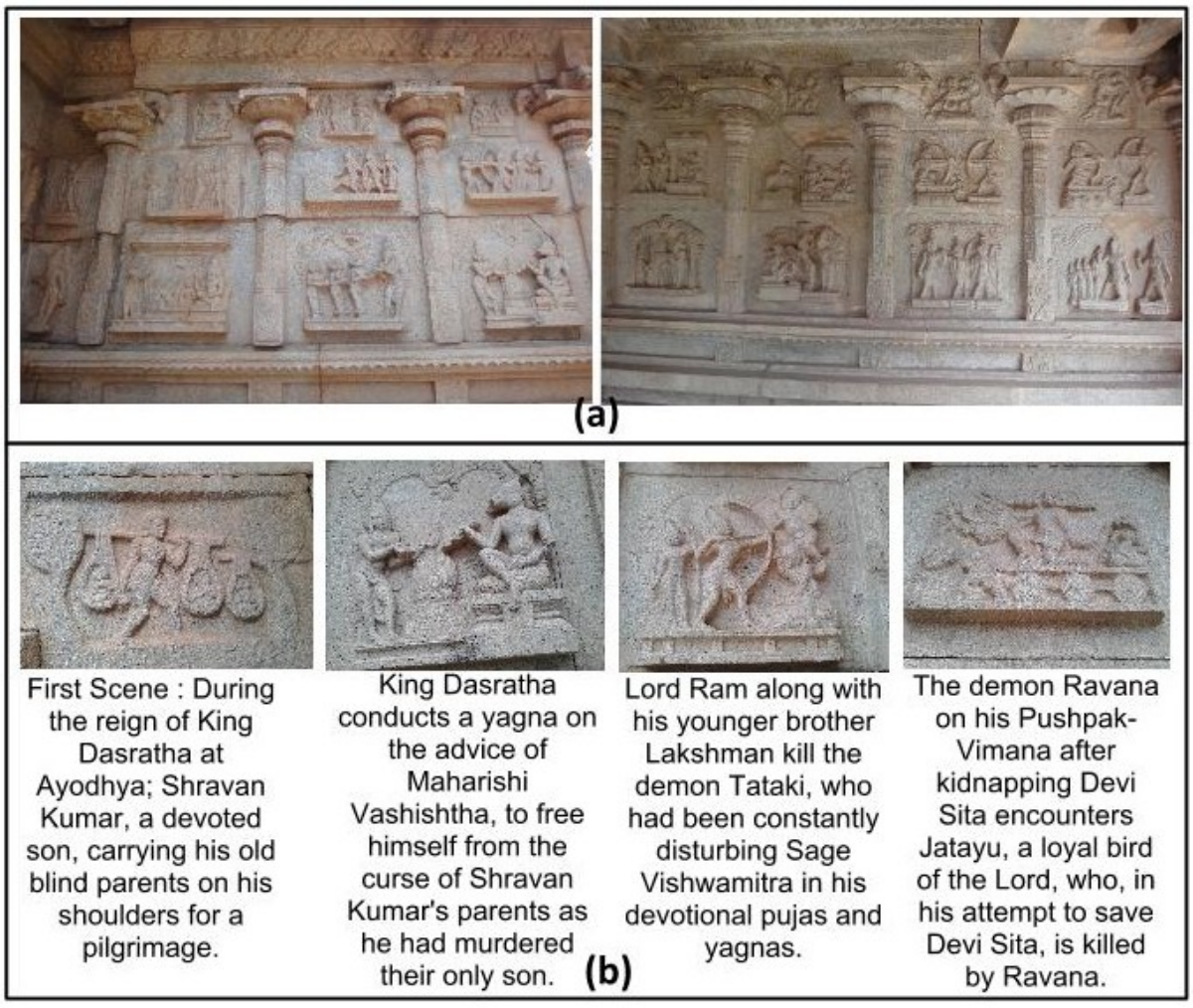


Figure 3.8: (a)Two walls of the main shrine at *Hazara Rama Temple, Hampi* depicting the story of *Ramayana* through stone-carvings in three tiers. (b)Four random mobile phone queries of *Ramayana* scenes with the corresponding annotation retrieved.

3.6 Summary

Preliminary annotations were collected from local tourist guides and the datasets were annotated with the efficient approach mentioned in chapter 2. Inorder to extend the utility of our mobile Heritage App, we want to reach out to the scholastic community in this domain of history and architecture. In this way, we seek to present more precise and detailed information to the end users.

We demonstrate a mobile vision app for scalable annotation retrieval at tourist heritage sites. The annotation retrieval pipeline is based on a robust BoW-based image retrieval and matching and works on an underlying database of annotated images. In the next chapter, we propose a slightly different approach to address the same problem so as the app has a smaller memory footprint without compromising on the speed and accuracy of retrieval results.

Chapter 4

Effective 2-word-phrase indexing for small memory footprint

This chapter further examines mobile image retrieval for *offline* use when a network connection is limited or not available. As discussed in chapter 2, the entire visual search index must reside on the mobile device itself. More specifically, we are interested in "instance retrieval", where the annotations associated with the images (e.g. building's name, object information) are returned by the query and not the images themselves. Figure 4.1 shows an example use case where mobile camera photos are used to identify buildings and landmarks without the need for a network connection.

Mobile image retrieval allows users to identify visual information about their environment by transmitting image queries to an online image database that has associated annotations (e.g. location, product information, etc) with the images. However, as argued in the previous chapter, this is reliant on a network connection to transmit and receive the query and retrieved information. Hence, a significant reduction in the visual index size is necessary. To achieve this, we describe a set of strategies that can reduce the visual index up to 60-80 $\times$ compared to a standard instance retrieval implementation found on desktops or servers. While our proposed reduction steps affect the overall mean Average Precision (mAP), they are able to maintain a good Precision for the top K results (P_K). We argue that for such offline application, maintaining a good P_K is sufficient. The effectiveness of this approach is demonstrated on several standard databases. A working application designed for a remote historical site is also presented. This application is able to reduce an 50,000 image index structure to 25 MBs while providing a precision of 97% for P_{10} and 100% for P_1.

4.1 Introduction

While our targeted instance retrieval does not need to store the images, the entire visual index structure needs to fit on a mobile device, ideally within a small footprint (e.g. 16-32MB). This small memory footprint serves two purposes. First, while mobile phones have up to 16-32GB of storage, this is mainly in the form of slower flash memory that is an order of magnitude slower than RAM. Having the entire index within tens of MBs makes it possible for use in a resident application on the phone's RAM. Second, this small size is inline with common practices for mobile applications; e.g. the iPhone average

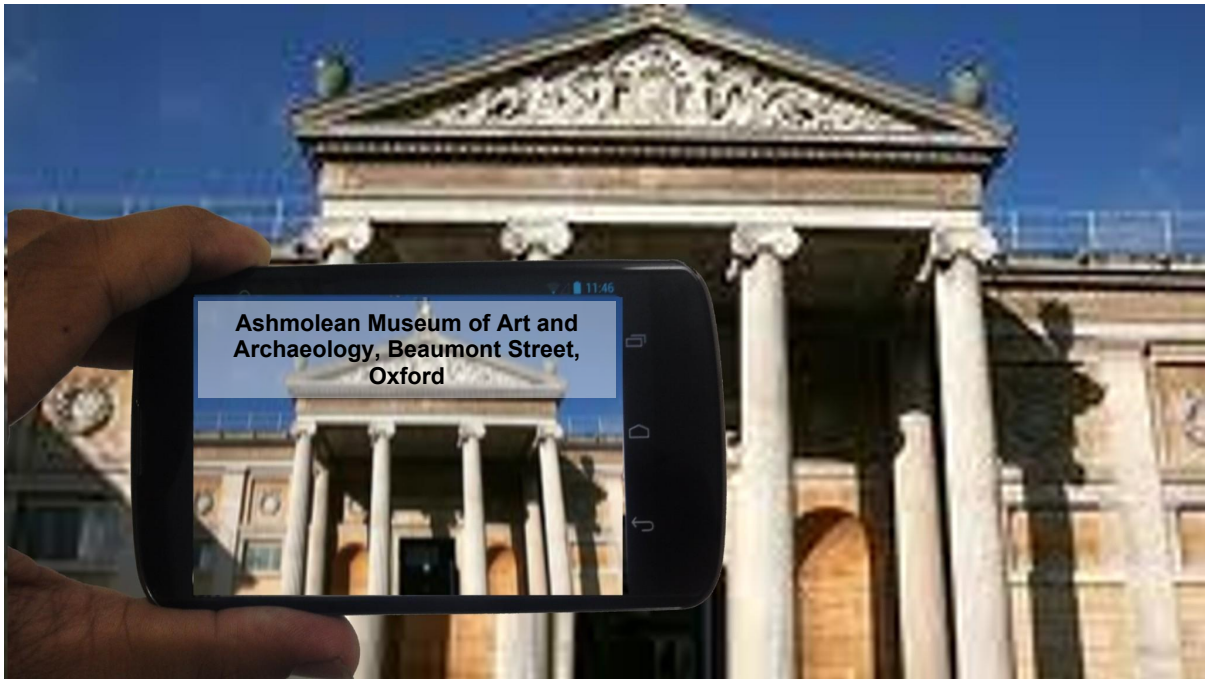


Figure 4.1: Example use of our method where landmark location can be performed offline by indexing an image collection within a small memory footprint for use on a mobile device.

application size is currently 23MB. Additionally, iPhone apps less than 50MB can be downloaded using 3G/4G, anything large must be downloaded using a wireless connection [1].

Contribution We introduce a series of pruning steps that can significantly reduce the index size and incorporate partial geometry in it. These include a simple vocabulary pruning based on word frequency, exploiting a 2-word-phrase based index structure, placing constraints on feature geometry and encoding the relative geometry of visual words. Each step is discussed in detail and a running example using the Oxford building dataset is used to show the effects on the index size and performance. While this reduction does come at a cost of a lower mAP, our reduction strategies do not adversely affect the mean precision at K (P_K). In fact, our collective strategies are able to reduce the visual index up to 60-80 $\times$ with virtually no change in P_{10}, P_5 and P_1. As we show, this is perfectly suited for mobile instance retrieval application. Higher precision for smaller K is important since, the eventual goal of the solution is to transfer the information from the top-K results to the query image.

Related Work We address the problem of instance retrieval in a challenging setting, where the entire computation must happen on the mobile device. This requires the entire database of information to reside on the device storage. Our approach in chapter 2 adopted simple pruning techniques with the existing Bag of words (BoW) framework to deliver a light-weight mobile app. In this chapter, we look to further compact the search index. In order to avoid any time-consuming post-processing, we want the index structure to effectively borrow the advantages of geometric verification.

Image recognition and retrieval primarily involves matching local image features which represent an image's appearance and geometry, to those in the database. Using Bag-of-Words approaches [34, 38, 44], scalability is achieved with the help of (i) quantization, which enables the efficient computation of similarity and (ii) order-less description of image features, making the representation invariant to many popular transformations. Variations of this approach have been motivated either towards building a compact and efficient search index or, a geometry induced search framework.

Decreasing the memory footprint of the search index has been earlier addressed targeting desktop environments [27, 29, 52]. Identifying a useful subset of training features that are robust and distinctive and using only these for specific retrieval tasks can achieve significant memory savings without loss in matching performance [49]. Hamming embedding techniques for feature representation in the form of binary strings have been introduced to efficiently work with smaller vocabularies [26]. Such techniques have also been employed for binary representation of global image features like GIST [48]. A popular alternative approach to inverted files in large-scale search is the min-hashing technique [10, 9]. Usage of Fisher Vectors [37] and vector of locally aggregated descriptors (VLADs) [28] for large-scale image search has been demonstrated with compact image vectors using joint optimization of dimensionality reduction and indexing for precise vector comparisons.

Words alone may not be meaningful to retrieve semantically related images. In such cases, one could use larger constructs (e.g. phrases) defined over multiple words as the basic primitive [32, 40]. A few attempts have introduced novel techniques which can help in phrases with index structures like min-hash [9, 54]. Introduction of geometry constraints into the phrase or index definition [36, 54, 56] often helps in improving the recall of the rare objects in large databases. However, with introduction of geometry, the search index size grows. Our approach attempts to achieve a compact geometry preserving indexing by efficiently utilizing the advantages of visual-phrase representation of images. We prune the largest possible 2-word-phrase space to design a compact solution.

4.2 Reducing the Index Footprint

We focus on retrieving images and/or associated information related to a specific instance/object, where the query can be from a widely varying imaging condition (like scale, view-point, illumination, etc) compared to those in the database. Instance retrieval literature has been dominated by the Bag-of-Words (BoW) method and its variations [10, 34, 38, 44].

In this section, we start by analysing the memory and storage requirements of the instance retrieval solution implemented as an inverted index search with a BoW representation. We experiment on the Oxford-5K dataset [38] to demonstrate the quantitative performance of various indexing strategies. This dataset contains 5,062 images from 11 different landmarks, each represented by 5 possible queries.

Local features (often SIFT vectors) are first extracted at interest points and quantized with the help of a precomputed vocabulary. With quantization, a 128-Byte SIFT representation gets converted to a compact visual word representation of 4 Bytes (or smaller depending on the vocabulary size). Visual

word representations are attractive for their invariance to the common transformations and compactness. Database images get represented as a histogram of visual words that are indexed with an appropriate index structure (often an inverted index). During the retrieval, SIFT vectors are extracted at interest points from the query image. They are quantized into visual words, and a set of images having most of these visual words in common, are retrieved from the database. Weighting visual words based on a TF-IDF measure has shown to improve the performance [44]. Even then, this list could contain many false positives. As such, the retrieved images are matched with the query image, and re-ranked based on the matching scores. Often, fitting a fundamental matrix or geometric verification (GV) of the two images is the basis for matching.

The choice of quantization scheme is a trade off between accuracy vs efficiency. Clustering with a flat k-means approach gives better results, but the quantization is linear in terms of the vocabulary. A vocabulary tree built with hierarchical k-means (HKM) [34] has logarithmic order and highly speeds up the vocabulary assignment, with small compromise on clustering results. Approximate k-means (AKM) [38] also has similar time and memory complexity as HKM. We choose to use HKM for our experiments.

For many instance retrieval tasks, the vocabulary size can be as high as 1M. If there are 5K images, the histogram representation could be as large as 5GB ($5\text{K}\cdot1\text{M}\cdot1\text{Byte}$). An efficient implementation with an inverted index can do this in around 110MB. An additional variable length encoding (VLE) technique could be used to further compress the index size. This, however, incurs additional decoding costs which may be undesirable on a mobile device. We report and compare the memory footprint without using VLE compression techniques for all our experiments. In addition, we need 2.3 GB memory to store the SIFT vectors with keypoint information, which are required during the final Geometric Verification (GV). Note, this does not count the storage of the image database. In Table 4.1, we present the space requirement as well as the retrieval performance of a desktop (BoW-D) implementation of a retrieval system on Oxford Building dataset. We show performance with and without GV. While GV improves the mAP and P_K, it requires significant computational resources, in addition, the resulting memory requirement is unacceptable for mobiles.

4.2.1 Baselines with Pruned Visual Words

We first reduce the memory requirement with two simplifications. First, for the problem of our interest, we do not store the images on the phone (we only need the annotation information or tags). This limits the storage to only the features and representations. Second, as done by other retrieval systems, instead of using SIFT vectors for GV, we match the visual word indices in the corresponding images. This reduces the memory requirement by a factor of 32 (i.e., $\frac{128}{4}$), without visible reduction in performance, as can be seen in Table 4.1 (see row:BoW-WGV).

Using such a vocabulary pruning, we reduce the vocabulary size from 1M to 750K, with only a minor reduction in performance as can be seen in Table 4.1. Baseline results of the pruned vocabulary (BoW-Pr and BoW-PrGV), are comparable to the original.

	Voc	Index	mAP	P_{10}	P_5	P_1
BoW-D	1M	110MB	.580	.75	.87	.88
BoW-D-GV	1M	2.4GB	.618	.79	.92	.93
BoW-WGV	1M	510MB	.616	.78	.92	.93
BoW-Pr	750K	80MB	.555	.75	.88	.91
BoW-PrGV	750K	480MB	.582	.77	.91	.92
Base-Phr	530K	58MB	.592	.78	.92	.93

Table 4.1: Baseline instance retrieval on the Oxford Buildings 5K dataset using the BoW framework. Last two rows reflect results with pruning and using 2-word-phrases respectively.

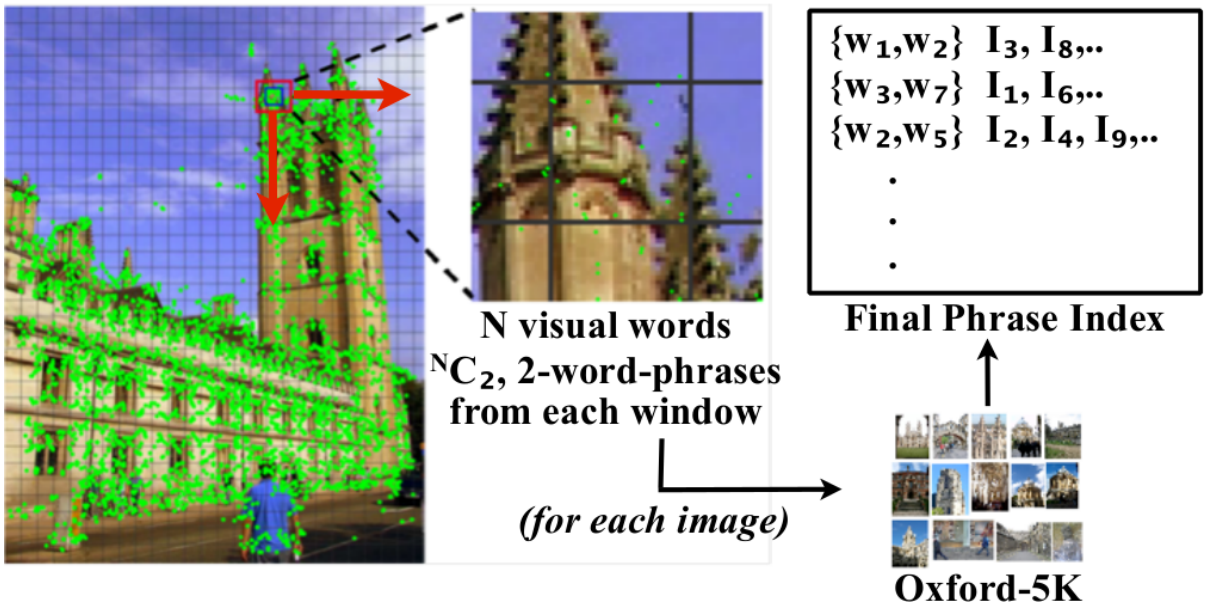


Figure 4.2: Building a 2-word-phrase index: Each image, represented by BoW quantized features, is processed to obtain phrases by dividing the image into "overlapping" grids and indexing 2-word combinations in each grid. In the final index, only phrases that index more than one image are retained.

4.2.2 2-word-phrase Search Index

Using a set of neighboring words (often referred to as phrases) is often more informative than isolated words. This has been attempted in various forms [4, 9, 40, 51]. Focus of most of the previous work has been on obtaining higher quantitative performance (i.e., mAP) during the retrieval. In this work, we first define a phrase as a pair of visual words, and show how that can help in the instance retrieval task.

A 2-word-phrase is detected from an object and is indexed in search records, if it occurs repetitively in images containing that particular object. Such phrases can be represented as $p_k = \{w_i, w_j\}$ such that w_i is in the neighborhood of w_j, and indexed in a similar manner to that of visual words. For implementation, we consider the neighborhood as a fixed size of 50 pixels. To avoid duplicate 2-word-phrases in the index structure, we constrain $w_i \leq w_j$. p_k is indexed in a hash-map with $\{w_i, w_j\}$ as a key.

The map data structure is typically implemented as binary search trees with logarithmic look-up time. Extracting higher order phrases increases the computational requirements during query processing. We find the additional computations are not affordable on low-processing devices. Further, indexing higher order features (≥ 3 words) is harder and do not always enhance the retrieval performance [53].

With a vocabulary size of 1M, the number of possible 2-word-phrases could be as large as 10^{12}, however, this does not happen in practice. We first find a set of phrases that are prominently present in the database. For example, phrases formed with a vocabulary as small as a 500K, one can obtain performance comparable to that of 1M visual words (see Table 4.1) without any explicit geometric verification. Even in this case, the subset of relevant phrases are computed based on the TF-IDF scores.

To speed up the computation, we divide the image into overlapping rectangular grids (see Figure 4.2) and choose combinations of visual word pairs from each grid. The overlap must be sufficient to ensure that true-positive 2-word-phrases are not missed out at grid-boundaries. A $40 - 50\%$ overlap is ensured between consecutive rectangular grids. Table 4.1 shows the comparison with traditional approaches. In row:Base-Phr, one can observe that the vocabulary is pruned to half the original size and the index requirement reduces by almost half to $58MB$. The mean average precision (mAP) increases by 1% as compared to baseline BoW method (BoW-D) and reduces by 2% as compared to BoW with geometric verification (BoW-D-GV). However, P_{10} and P_5 are comparable with the latter. Since we are motivated to look for the most similar image and not bother about the rank-list, the drop in mAP is ignored.

4.3 Geometry-aware 2-word-phrases

For our problem of instance retrieval, such as location recognition or product search, further constraints on the geometry can be assumed. For example, the variation of depth within an object will likely be small, even if the object/product is not planar. In this section, we demonstrate how the geometry can be effectively used in selecting a much smaller subset of phrases for our instance retrieval problem. In general, we constrain the phrases selected, based on (i) the scale of keypoints at the visual word location, (ii) the relative placement of the words, and (iii) the possibility of it contributing reliably for a geometric verification. We compare the results with the standard desktop version of BoW with GV (BoW-D-GV), and with the state-of-the-art image search techniques with compact representations: the Fisher Vectors (FV) and Vector of locally aggregated descriptors (VLADs) [29]. Note we are using the basic implementation for VLADs in [29], recent improvement gains in results have been demonstrated [2].

4.3.1 Scale-aware constraints

We define phrases only in a small neighborhood (50 pixels in practice) and assume both the visual words are at the same scale. This is mostly true except for situations of serious depth discontinuity. However, for situations like product search or location search, this is a reasonable assumption. Phrases

	Voc	Index	mAP	P_{10}	P_5	P_1
ScA-Phr	462K	41MB	.581	.78	.92	.93
SpA-Phr	280K	36MB	.579	.78	.92	.93
Scp-Phr	250K	28MB	.571	.78	.91	.92
SV-Phr	180K	16MB	.567	.77	.90	.91
BoW-D-GV	1M	2.4GB	.618	.79	.92	.93
FV	64	4.5MB	.298	.50	.66	.76
VLAD	256	23MB	.309	.49	.63	.69

Table 4.2: Results on using the Geometry aware 2-word-phrases. ScA-Phr, SpA-Phr, Scp-Phr and SV-Phr Phr refer to the the 2-word-phrase index methods with scale-aware, space-aware, both scale-space aware and spatial validity constraints respectively. The last three rows refer to the state-of-the-art Bag-of-words (as BoW-D-GV in Table 4.1), Fisher Vectors and VLAD approaches [29].

defined using words with the same scale are referred to as scale-aware phrases. This also helps in selecting phrases that will be reliable for viewpoint variations.

During feature extraction, SIFT keypoints are detected at scale-space extrema points using the DOG-space pyramid, constructed at scale-levels that are multiples of 2. We separate scale according to log-scale $(log(s_i))$ value corresponding to the feature keypoints detected at s_i scale. Two visual words with keypoint scales s_i and s_j, are combined into a 2-word-phrase, if $|(log(s_i) - log(s_j))|$ is less than a particular threshold τ. For our implementation we use τ as *3*. The separation of scale allows to enlarge our rectangular window size to $70 - 100$ pixels as the number of 2-word-phrase combinations get reduced significantly. This accommodates more true-positives and reduces the probability of false-positives. In Table 4.2, with scale-aware phrases (ScA-Phr) the search index is reduced with a further pruned vocabulary, while precision P_{10}, P_5 and P_1 are preserved.

4.3.2 Space-aware constraints

Phrases capture larger geometry of the configuration of visual words. We constrain the geometry of occurrence of two visual words further by encoding the relative placement of visual words in a neighbourhood. We embed a loose spatial constraint using just 2-bits of information. In a phrase p_k defined by $\{w_i, w_j\}$, where $w_i \leq w_j$, we choose w_i as the origin and determine in which rectangular quadrant w_j lies in (See Figure 4.3). Depending on the quadrant where w_j lies, a two-bit value of $q_{ij} = 0, 1, 2, or, 3$, the 2-word-phrase key is modified as $\{p_k, q_{ij}\}$ i.e. $\{w_i, w_j, q_{ij}\}$.

With just two-additional bits per index key, we further prune the vocabulary size and reduce the phrases from the search index. In Table 4.2, row:SpA-Phr corresponding to space-aware phrases, shows a decrease in the size of search index. Using both the scale and space constraints (row:Scp-Phr) together, we achieve a search index structure as small as $28MB$ with a vocabulary pruned to $250K$.

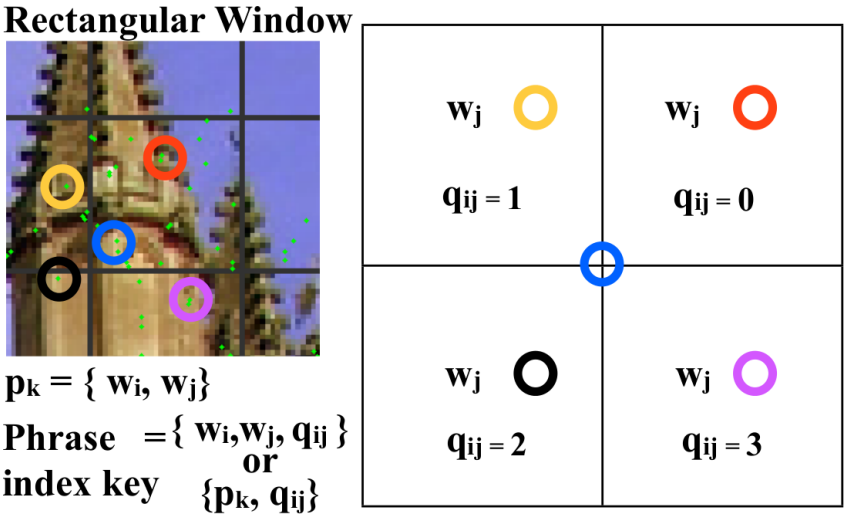


Figure 4.3: Space aware constraint is enforced with a two bit information q_{ij} along with the 2-word-phrase $\{w_i, w_j\}$ as shown.

4.3.3 Spatial Validity Constraints

Motivated by the approach presented in [49], we retain useful features by validating the training features in each image using an unsupervised pre-processing. Useless features that come from occlusions and unstable keypoints are likely to exist in only a single image, while useful features are found in more than one image of the same object. To identify useful features, each image is used as a query and geometric verification is performed on the top M retrievals based on visual word matching. The features that score as inliers in this procedure are considered useful and hence retained. Other features are discarded. Using $M = 100$, we observe that this brings down the average number of features per image by 90% (See Figure 4.4).

We now prune 2-word-phrases that have a component visual feature rendered useless by the above technique. Hence, the phrases obtained by the combination of space and scale aware constraints are further processed. This brings down the vocabulary size to $160K$ and the search index is reduced to $16MB$. As shown in Table 4.2 (row:SV-Phr), phrases constrained with spatial validity further compromise the mAP over the conventional BoW-D-GV approach but help in significant reduction of index size.

4.3.4 mAP vs P_K

Image retrieval research is often focused on achieving a superior mAP of the retrieved images. This is achieved by pulling out rare images or images which contain the object or location of interest in only a small part of the image. This is relevant when the objective is to retrieve from a "casual" database of images. We are working on a database of images with annotations (which are often prepared and specially annotated by a user). Given a specific view of a product or building, we are not interested in

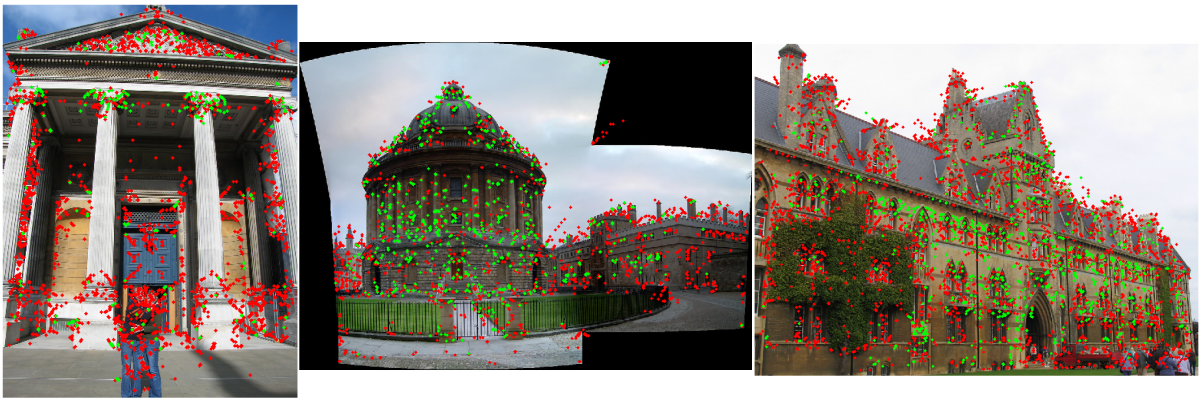


Figure 4.4: Spatially validated features in example images. The red dots represent the rejected features after geometric validation based pre-processing, while green dots are the useful features. We can observe the unstable features due to occlusion are removed.

getting all the images of the same object. Instead, we desire a similar view in a reliable manner and possibly transfer the annotation to the query image. This needs highly similar images coming at the top of the retrieved list. Our retrieval process is designed to retain the highly similar images at the top of the list. This is done by (i) using the scale information while defining phrases and (ii) spatial constraints preferred images with similar view angles to be at the top of the list.

4.4 Results and Discussions

We demonstrate that the method of using 2-word-phrases in our search index, significantly reduces the size of the Inverted look-up index. By identifying 2-word-phrases that inherit robust geometrical constraints, we are able to further reduce the search index by 3-4 times. In the previous sections, we make use of Oxford-5K as our primary evaluation dataset. We now show results on three other datasets.

Evaluation Criteria: Image retrieval systems measure the performance in terms of the mAP. Average precision is the area under the precision-recall curve. This is computed for each query based on the ranked list of retrieved images, and the average performance on all queries give us the mAP. We are interested in search applications that are not concerned with the ranked list, but only the top image. Hence, although we take care of sharp drop in AP, only the Precision at K (P_K) matters. Hence, we measure the mean Precision at $10, 5, 1$ (P_{10}, P_5, P_1) to evaluate our performance.

4.4.1 Performance on Multiple Datasets

Paris Buildings [39] : This dataset has 6,386 high-resolution (1024×768) images of 11 iconic landmarks from France. Five query images are taken from each of the 11 buildings to give a total of 55 queries. The images are marked by architecturally distinct landmarks unlike identical buildings in Oxford-5K, which creates a difference in the vocabulary used for quantization. With a vocabulary size

Figure 4.5: Random Sampling of images from each of the four datasets used for evaluation: Oxford Buildings, Paris Buildings, UkBench, and Heritage structures.

Dataset	#Images	#Queries
Oxford Buildings	5,062	55
Paris Buildings	6,383	55
UkBench Dataset	10,200	10,200
Heritage Structures	50,292	40

Table 4.3: Number of images in each dataset and the number of queries used for evaluation.

of $1M$, we observe that the mAP is affected with our 2-word-phrase indexing technique. However, we were able to reduce the memory footprint significantly, compared to BoW approach with SIFT-based GV (BoW-D-GV).

UkBench dataset [34] : This dataset has 10,200 medium-resolution (640×480) images with 2,550 groups of 4 images each. Each group has images of a particular object from different viewpoint and angle variations. Since, the objects are very distinct, this dataset has large variations in appearance. A vocabulary of size $1M$ was built using features from all the objects. The evaluation criteria here is P_4 and P_1. Each image in the dataset is taken as a query image and performance is measured. It was observed that on average, we usually get more than 2 out of the 4 positives in the top-4. The size of the search index is reduced by 75 times, compared to BoW-D-GV.

Heritage Dataset : We introduce a novel data set of images from a world heritage site. This dataset has 50,292 medium-resolution (480×360) images of specific buildings, architectural details, interiors

Dataset	Method	Voc	Index	P_{10}*	P_1
Paris (6,086)	BoW-D-GV	1M	1.9GB	.98	1.0
	Base-Phr	300K	51MB	.91	.95
	Geom-Phr	200K	34MB	.91	.98
ukbench (10,200)	BoW-D-GV	1M	1.2GB	.76	.92
	Base-Phr	300K	22MB	.65	.88
	Geom-Phr	200K	16MB	.68	.91
Heritage (50,292)	BoW-D-GV	1M	1.6GB	.97	1.0
	Base-Phr	350K	37MB	.96	1.0
	Geom-Phr	200K	25MB	.97	1.0

Table 4.4: Performance on the remaining three datasets. *: For the Ukbench dataset, we evaluate P_4 instead of P_{10}. Base-Phr refer to baseline 2-word-phrases and Geom-Phr shows results with geometry-aware 2-word-phrase indexing.

and exteriors of heritage monuments, etc. (See Figure 4.5, last two rows). Four query images are taken from each of the 10 distinct and iconic monumental structures we identified to evaluate on, giving us a total of 40 queries. This dataset has similarities with the Oxford-5K in terms of the identical buildings and structures, marked by a unique style of architecture. We measure performance in terms of precision at 10, 5, 1. We reduce the search index size significantly, however, P_K is preserved.

4.4.2 Analysis

Selecting a phrase or co-occurrence of visual words from a query image to match with a similar co-occurrence in the database brings in spatial constraints during the matching procedure. Along with further addition of loose-geometry to the pair of visual words in a phrase, the constraints become stronger in order to avoid false matches.

With the help of 2-word-phrases as a key in the search index, our method tries to achieve the same precision at ranks 10, 5 and 1 as obtained by the conventional BoW-based approach with the GV post-processing. Matching spatially and geometrically constrained 2-word-phrases partially incorporates the advantages of GV. In Figure 4.6, we can observe that while the raw individual visual word matches include a lot of false-positives, on matching 2-word-phrases, we mostly have true-positive matches.

We also observe that the 2-word-phrases help in significantly reducing the size of the search index. This can be realised if we understand that only stable feature keypoints are preserved and indexed in the search records. Lone keypoints that cannot be paired with others in the chosen neighborhood, as well as those set of pairs that occur in single images, are rejected. This results in pruning of the original vocabulary. As the vocabulary size decreases, the memory footprint for the search index also decreases. This affects the mAP. However, we observe that the Precision at $10, 5, 1$ remain mostly unaffected (See Figure 4.7). This is because, while the ranked-list is affected with smaller search index, the top results are in general, preserved.

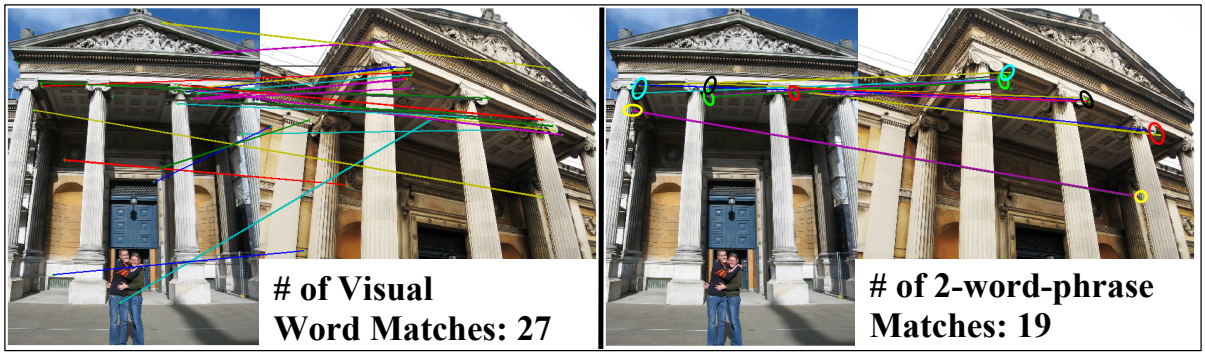


Figure 4.6: Visual word vs. 2-word-phrases matches between two images of the same object captured from very distinct view points. While word matches require a geometric verification to remove false positive matches, most of the matched 2-word-phrases are true. A few two-word-phrases-matches are highlighted with same-colored ellipses around the keypoint pairs.

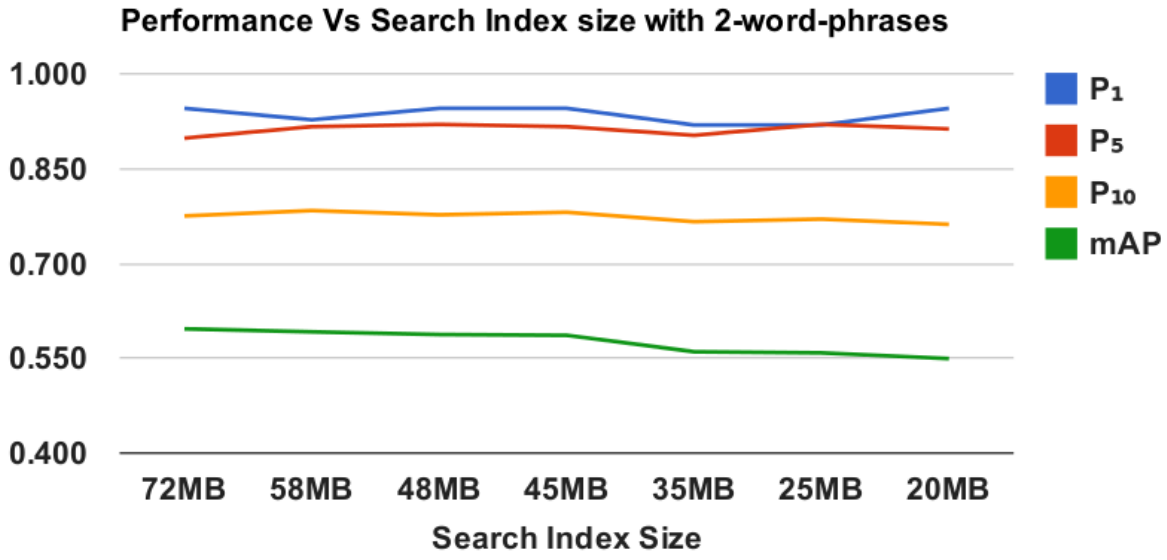


Figure 4.7: Graph of Performance Vs Index size with the 2-word-phrases indexing. The mAP decreases with size of the search index. However, Precision at 10, 5, 1 remain largely unaffected.

Event	**Time**(in seconds)
SIFT keypoint Detection	0.250
Extracting SIFT descriptors	0.270
Quantizing to Vocabulary	0.010
Compute 2-word-phrases	0.470
Index Search	0.080
Total	1.080

Table 4.5: Average computational time analysis on a mobile phone with 1GHz processor. Feature extraction and constructing 2-word-phrases take up significant time.

Figure 4.8: An application of the compact search index structure for tourism. Mobile phone localizes and gives details of what it sees without the help of external network. The first row shows usage at different locations and monuments; the second row demonstrates the robustness of the approach to view-changes of a particular monumental structure.

4.4.3 Mobile App

We now discuss an implementation for a mobile instance search on a mid-end mobile phone. Our application provides details of the monuments based on the image captured on a mobile phone. The app is implemented for Android devices and designed to allow a tourist point his mobile at an object and know its details without using a network connection. Examples of the recognition is shown in Figure 4.8.

The app is designed for the heritage data set discussed in the section 4.4.1. The search index is built on a dataset with 50,292 medium resolution (480×360) images, which amounts to $5.3GB$. Note that the images are never stored on the mobile devices. We extract SIFT features from all images and use hierarchical k-means to build a corresponding visual vocabulary of size $100K$. The memory occupied by the vocabulary tree during query processing is $17MB$. With our 2-word-phrase indexing technique, we get a compact search index of $25MB$. These along with the annotations for the database images are all that we store as application data on the mobile device.

(a) At the Maha Mantapa.

(b) At the Kudregombe Mantapa.

(c) At the Gejjala Mantapa.

(d) At the Sabha Mantapa.

(e) At the Raj Gopuram.

(f) At the Function Hall.

Figure 4.9: Successful field testing cases at Vittala Temple, Hampi.

The heritage site is marked by architecturally similar monumental structures and towers spread over a large area. We did real-time testing of our app at this site and analyzed how our solution performs in terms of correctness of retrieved annotations, memory usage and the speed of annotation delivery. The app successfully returned correct annotations at a wide-variety of locations and for different monuments. On average, the annotation was delivered within a second or two (See Table 4.5). Since, this is a tourist location, we did face failure cases, when a large part of the monument was occluded by crowd, however, overall we found the app to be successful at its task.

4.5 Field Testing at Vittala Temple, Hampi, India

We conducted an extensive field testing for a particularly famous and interesting temple complex at the World Heritage Site of Hampi, Karnataka, India [21]. Using the Batch Annotation Approach of chapter 2, we annotated a dense curated dataset of 4729 images captured from the temple keeping with a focus on the distinct structures, for which we felt the necessity of presenting auto-annotations on a mobile instance retrieval app. On a whole, 51 distinct structures were annotated within the temple complex.

We were able to get successful annotations from varying view points and for different monuments within the temple complex. We also experimented in cloudy and sunny conditions and got satisfying results. A feedback session was conducted by allowing random tourists to use the app on our device. We received a positive feedback overall, a video of which can be seen here [1]. A set of successful test cases, captured during the field trial are shown in Figure 4.9.

4.6 Summary

This paper presents an indexing strategy for reducing the memory footprint for instance retrieval along with exploiting the spatial geometry of 2-word-phrases. Precision at K is preserved even with significant reduction in the size of search index. We successfully demonstrate the usage of 2-word-phrases as key in the inverted search index and how this helps in pruning the vocabulary, compacting the search index as well as inducing geometric constraints within the index structure. This facilitates successful application in low-memory, slow-processing mobile environments.

[1] http://www.youtube.com/watch?v=P6oz597xmXs

Chapter 5

Conclusions

This thesis presents an effective solution to perform instance retrieval on a mobile phone and use this to recognize and annotate a heritage monument in the scene. The methods proposed here, focus on the limited processing and memory environment of an offline mobile phone. A robust mobile app, aptly named as "Heritage App" is demonstrated for two famous heritage sites in India. The problem, which seemed a herculean task for a mobile phone at first, was boiled down using simple and efficient indexing mechanisms and effectively prune the visual vocabulary, building upon the basic Bag of Words (BoW) framework and significantly compacting the search index structure. As discussed in Chapter 4, there have been other attempts at reducing the memory footprint for instance retrieval particularly targeting large-scale or web-scale image datasets [29]. While these approaches have been designed for desktop or server platforms, there is a wide and interesting scope to modify the baseline approaches for mobile platforms as well. This domain opens up a new dimension of solutions for mobile phones that may prove as effective and scale up to a large collection of database images.

The techniques proposed in this work focused on recognition of 3D monuments or buildings, but are intuitive to work on a variety of other objects like retail products, printed media, etc. This widens the scope of offline instance retrieval apps on mobile phones that can be deployed using the same pipeline demonstrated here. Further, the solution may be incorporated into other applications that need localisation on the mobile phone, say for example, to help in 3D pose estimation. Indirect calibration techniques that need 2D-2D-3D matching to estimate the pose of a camera with respect to a 3D object, can exploit the offline mobile instance retrieval techniques mentioned in this thesis to solve the 2D-2D matching problem.

There is a wide scope for possible future works in terms of more compact index structures that can lead to lighter apps with lesser memory footprints on a mobile phone. The techniques used in this work to port a heavy-weight computer vision algorithm to the highly constrained mobile environment could be exploited in other similar tasks. This opens up a problem of designing and developing configurable computer vision apps on mobile phones. One can also innovate in the pipeline by introducing efficient and robust interest point detectors and descriptors that are faster than SIFT and not compromise the accuracy. Recent algorithms like FAST, ORB, FREAK, BRISK, etc show promising results, but are not

as invariant and robust as the state-of-the-art SIFT detectors and descriptors. Further, the apps can be designed for effective colloboration across multiple phones so that people can read the relevant feedbacks, comments and recommendations. User logs from mobile phones can give a decent knowledge on the mobile use patterns, which can be useful information from a developers' point of view.

Another takeaway from the thesis is the efficient batch annotation approach in Chapter 2. It presents a decent web-based framework to richly annotate a large dataset of images. We have shown the importance of such a curated dataset in view of our mobile apps that use offline instance retrieval for describing a query monument captured in heritage sites. The annotated images help in organized browsing of large photo collections and may also be used for more efficient text-based image retrievals.

Related Publications

- Jayaguru Panda, Michael S. Brown and C. V. Jawahar, "Offline Mobile Instance Retrieval with a Small Memory Footprint", In **ICCV - 2013** at Sydney, Australia
- Jayaguru Panda and C. V. Jawahar, "Efficient and Rich Annotations for Large Photo Collections", In Proceedings of *Asian Conference on Pattern Recognition (ACPR), 2013* at Naha, Japan
- Jayaguru Panda, Shashank Sharma and C. V. Jawahar, "HeritageApp: Annotating Images on Mobile Phones", In Proceedings of the *Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP), 2012* at Mumbai, India

Bibliography

[1] ABIresearch http://www.abiresearch.com/press/average-size-of-mobile-games-for-ios-increased-by-. accessed: 2-April-2012.

[2] R. Arandjelović and A. Zisserman. All about VLAD. In *CVPR*, 2013.

[3] G. Bernd, V. Chandrasekhar, D. M. Chen, N. man Cheung, R. Grzeszczuk, Y. Reznik, S. Tsai, G. Takacs, and R. Vedantham. Mobile visual search. *IEEE SPM*, 2011.

[4] Y. Cao, C. Wang, Z. Li, L. Zhang, and L. Zhang. Spatial-bag-of-features. In *CVPR*, 2010.

[5] V. Chandrasekhar, D. M. Chen, Z. Li, G. Takacs, S. S. Tsai, R. Grzeszczuk, and B. Girod. Low-rate image retrieval with tree histogram coding. In *MobiMedia*, 2009.

[6] V. Chandrasekhar, Y. Reznik, G. Takacs, D. Chen, S. Tsai, R. Grzeszczuk, and B. Girod. Quantization schemes for low bitrate compressed histogram of gradients descriptors. In *CVPR Workshops*, 2010.

[7] V. Chandrasekhar, G. Takacs, D. M. Chen, S. S. Tsai, Y. Reznik, R. Grzeszczuk, and B. Girod. Compressed histogram of gradients: A low-bitrate descriptor. *IJCV*, 2012.

[8] D. M. Chen, S. S. Tsai, V. Chandrasekhar, G. Takacs, J. P. Singh, and B. Girod. Tree histogram coding for mobile image matching. In *DCC*, 2009.

[9] O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *CVPR*, 2009.

[10] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *BMVC*, 2008.

[11] A. Deshpande, S. Choudhary, P. J. Narayanan, K. K. Singh, K. Kundu, A. Singh, and A. Kumar. Geometry directed browser for personal photographs. In *ICVGIP*, 2012.

[12] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google's image search. In *ICCV*, 2005.

[13] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981.

[14] P. Föckler, T. Zeidler, B. Brombach, E. Bruns, and O. Bimber. Phoneguide: museum guidance supported by on-device object recognition on mobile phones. In *Mobile and ubiquitous Multimedia*, 2005.

[15] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view stereo for community photo collections. In *ICCV*, 2007.

[16] J. Graham and J. J. Hull. Icandy: a tangible user interface for itunes. In *CHI '08 extended abstracts on Human factors in computing systems*, 2008.

[17] J. Hays and A. A. Efros. Scene completion using millions of photographs. *SIGGRAPH*, 2007.

[18] J. Hays and A. A. Efros. Im2gps: estimating geographic information from a single image. In *CVPR*, 2008.

[19] N. Henze, T. Schinke, and S. Boll. What is that? object recognition from natural features on a mobile phone. In *Workshop on MIRW*, 2009.

[20] http://flow.a9.com.

[21] http://hampi.in/vittala temple.

[22] https://betalabs.nokia.com/trials/nokia-point-and find.

[23] http://www.google.com/mobile/goggles.

[24] http://www.kooaba.com.

[25] http://www.vijayanagara.in.

[26] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008.

[27] H. Jégou, M. Douze, and C. Schmid. Packing bag-of-features. In *ICCV*, 2009.

[28] H. Jegou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[29] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *PAMI*, 2012.

[30] R. Ji, L.-Y. Duan, J. Chen, H. Yao, T. Huang, and W. Gao. Learning compact visual descriptor for low bit rate mobile landmark search. In *IJCAI*, 2011.

[31] R. Ji, L.-Y. Duan, J. Chen, H. Yao, Y. Rui, S.-F. Chang, and W. Gao. Towards low bit rate mobile visual search with multiple-channel coding. In *ACM Multimedia*, 2011.

[32] Y. Jiang, J. Meng, and J. Yuan. Randomized visual phrases for object search. In *CVPR*, 2012.

[33] D. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[34] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[35] J. Panda, S. Sharma, and C. V. Jawahar. Heritage app: annotating images on mobile phones. In *ICVGIP*, 2012.

[36] M. Perd'och, O. Chum, and J. Matas. Efficient representation of local geometry for large scale object retrieval. In *CVPR*, 2009.

[37] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *CVPR*, 2010.

[38] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[39] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008.

[40] S. Romberg, M. August, C. X. Ries, and R. Lienhart. Robust feature bundling. In *Advances in MIP*, PCM, 2012.

[41] G. Schroth, R. Huitl, D. Chen, M. Abu-Alqumsan, A. Al-Nuaimi, and E. Steinbach. Mobile visual location recognition. *IEEE SPM*, 2011.

[42] I. Simon and S. M. Seitz. Scene segmentation using the wisdom of crowds. In *ECCV*, 2008.

[43] I. Simon, N. Snavely, and S. M. Seitz. Scene summarization for online image collections. In *ICCV*, 2007.

[44] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[45] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski. Finding paths through the world's photos. *SIGGRAPH*, 2008.

[46] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH*, 2006.

[47] G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W.-C. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod. Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In *ACM MIR*, 2008.

[48] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *CVPR*, 2008.

[49] P. Turcot and D. G. Lowe. Better matching with fewer features: The selection of useful features in large database recognition problems, 2010.

[50] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *ISMAR*, 2008.

[51] Z. Wu, Q. Ke, M. Isard, and J. Sun. Bundling features for large scale partial-duplicate web image search. In *CVPR*, 2009.

[52] X. Zhang, Z. Li, L. Zhang, W.-Y. Ma, and H.-Y. Shum. Efficient indexing for large scale visual search. In *ICCV*, 2009.

[53] Y. Zhang and T. Chen. Efficient kernels for identifying unbounded-order spatial features. In *CVPR*, 2009.

[54] Y. Zhang, Z. Jia, and T. Chen. Image retrieval with geometry-preserving visual phrases. In *CVPR*, 2011.

[55] Y.-T. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T.-S. Chua, and H. Neven. Tour the world: building a web-scale landmark recognition engine. In *CVPR*, 2009.

[56] W. Zhou, Y. Lu, H. Li, Y. Song, and Q. Tian. Spatial coding for large scale partial-duplicate web image search. In *ACM Multimedia*, 2010.