Heritage App: Annotating Images on Mobile Phones

Jayaguru Panda
jayaguru.panda@research.iiit.ac.in

Shashank Sharma
shashank.sre.iiit@gmail.com

C.V. Jawahar
jawahar@iiit.ac.in

Center for Visual Information Technology (CVIT), IIIT-Hyderabad, India

ABSTRACT

In this paper, we demonstrate a computer vision application on mobile phones. One can take a picture at a heritage site/monument and obtain associated annotations on a mid-end mobile phone instantly. This 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 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.

Keywords

Image Retrieval, Bag of Words, Annotation, Mobile Vision

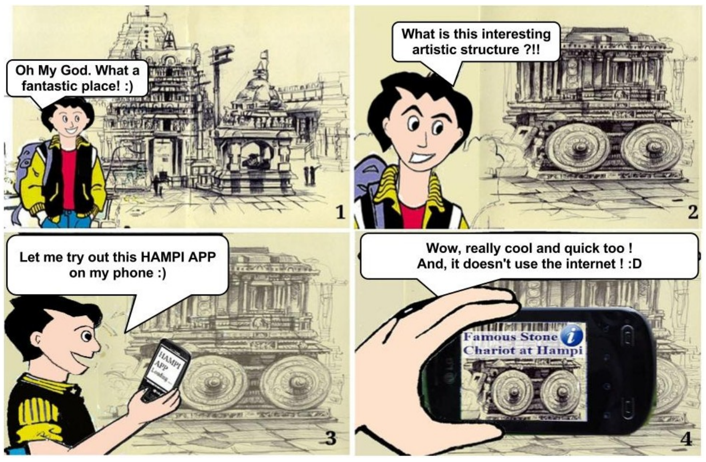
Oh My God. What a fantastic place! :)
1
What is this interesting artistic structure ?!!
2
Let me try out this HAMPI APP on my phone :)
HAMPI APP loading...
3
Wow, really cool and quick too ! And, it doesn't use the internet ! :D
Famous Stone Chariot at Hampi
4

Figure 1: A typical use case for our mobile app, displaying auto-annotation at a temple in Hampi.

1. INTRODUCTION & RELATED WORK

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 [18], product search [1], tourism [11], business [2], entertainment [12] and other generic image recognition apps like Amazon Snaptell [1], Nokia point and find [3], Google Goggles [4], Mobile Visual Search [5]. 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 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 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)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICVGIP '12, December 16-19, 2012, Mumbai, India
Copyright 2012 ACM 978-1-4503-1660-6/12/12 ...$15.00.

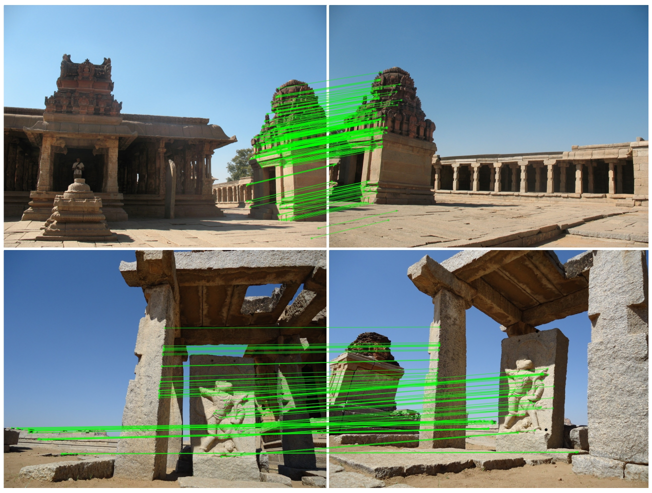

Figure 2: An example of spatial verification on Hampi images, where a Fundamental Matrix is fit across two images.

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.* [10] 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 [9, 14, 8, 6, 7, 15]. 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 [11, 20, 21, 13]. 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.* [13] and Fockler *et al.* [11] 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.

2. INSTANCE RETRIEVAL WITH BOW

The computer vision community works on two kinds of retrieval problems : (i) Categorization, or, object class recognition, which concerns with the retrieval of a category/class of objects (for ex, "houses"). (ii) Instance, or object retrieval, which deals with retrieval of a particular object (for ex, "my house"). We address the problem of Instance Retrieval, similar to the retrieval problem posed for the standard Oxford Buildings dataset [17]. The web-scale landmark recognition system [22] 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 [19] 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 [19]. 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 2) over the point correspondences between query and retrieved images. This typically improves the mean Average Precision (mAP) of the retrieval process in instance retrieval [17]. 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. 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 600 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. 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 [17] to demonstrate the advantages of the enhanced BoW.

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 [16] 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.

3.2 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 $\mathrm{V}_i = \mathrm{V}_j$. 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 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.

3.4 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

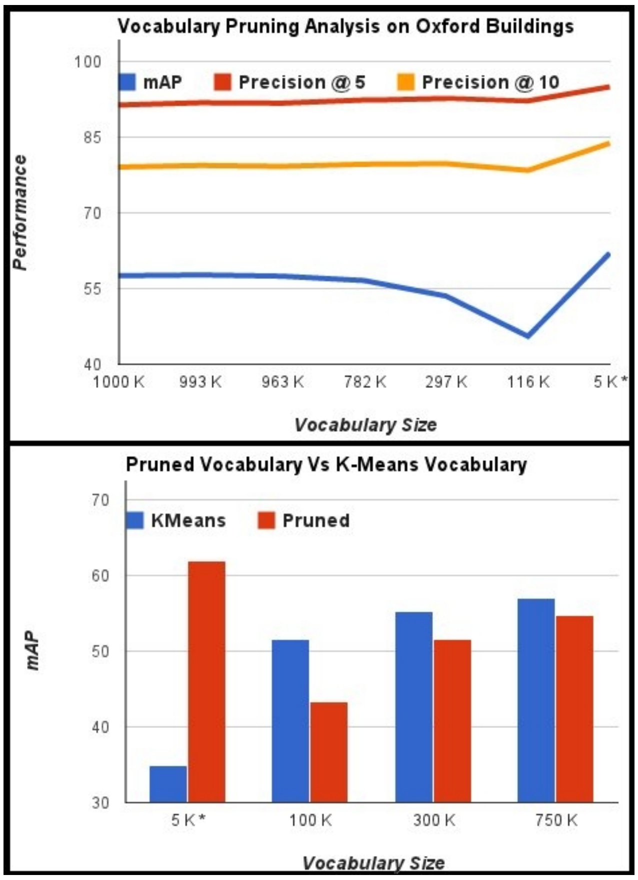

Figure 3: Performance analysis for different pruned vocabularies. *: In the supervised method, we get one pruned vocabulary of size 5K, that performs better than others. The other sizes >100K were obtained in the unsupervised method.

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 $\tau_L >= n_i$ or, if $n_i >= \tau_H$, 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 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).

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 (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.

3.5 Database Pruning

In section 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 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 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.

	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 1: Database Pruning Results on Oxford Buildings and Golkonda datasets

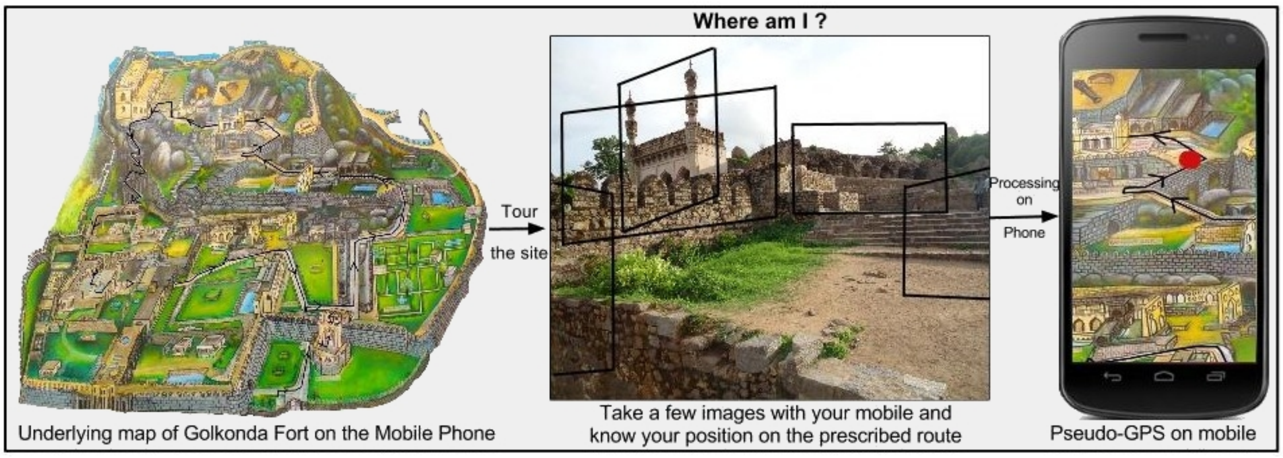
Where am I ?
Tour
the site
Processing
on
Phone
Underlying map of Golkonda Fort on the Mobile Phone
Take a few images with your mobile and
know your position on the prescribed route
Pseudo-GPS on mobile

Figure 4: Usage of image-based Pseudo-GPS to find the current location on the prescribed route of Golkonda Fort

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.

4.1 Scene and Object Annotations

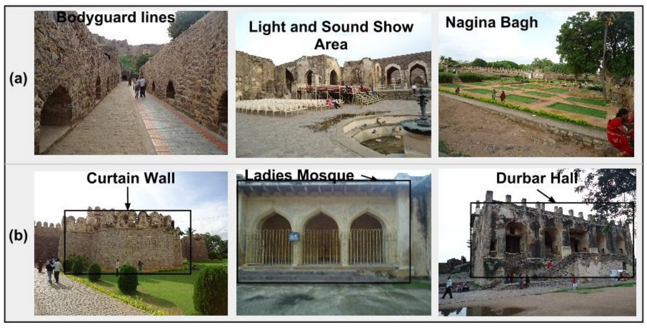

Figure 5: Sample images with (a)Scene and (b)Object annotations

We choose to classify annotations into two different categories: scene-based and object-based (See Figure 5). 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.

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 4), 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.

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

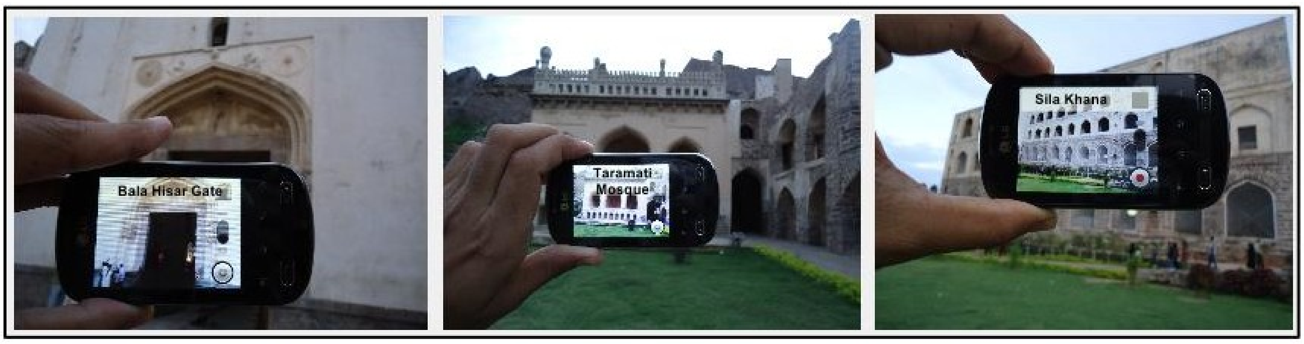

Figure 6: Mobile Heritage App Illustration for 3 different scenes at Golkonda Fort

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.

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 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.

	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 2: Time Analysis for the Heritage App on a mobile phone with 600 MHz processor and 512 MB RAM

5.1 Dataset and Annotations

We demonstrate our work 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 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.

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.

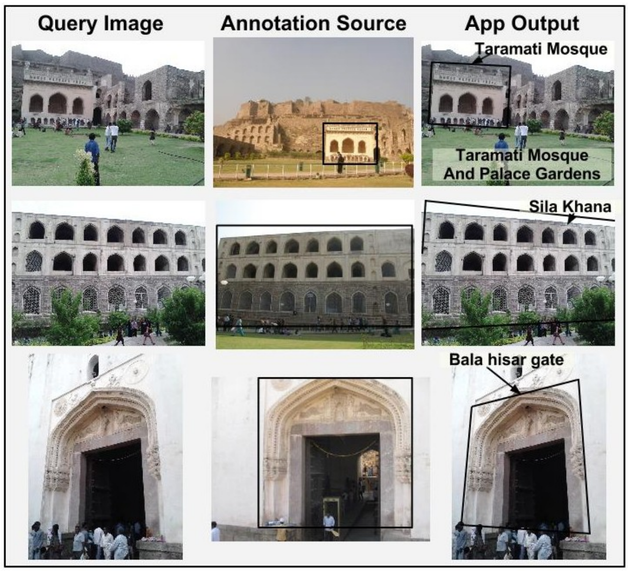

Figure 7: Annotation retrieval for three sample query images at Golkonda Fort.

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 7) 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). We also observe the time taken for processing a query image on the mobile device (See Table 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 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 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.

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

Table 3: Performance Analysis: Mean Annotation Accuracy for Heritage App for Golkonda Fort and Hampi Temples

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

Our mobile app performs poorly if the query image has crowd presence (See Figure 8). 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.

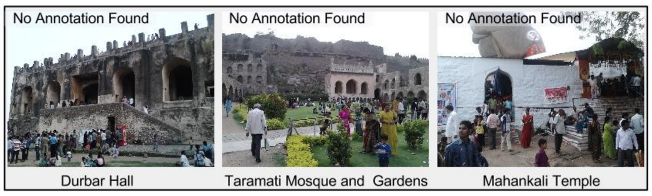

Figure 8: Query Images with crowd presence, for which our Heritage App fails in retrieving correct annotations from the database.

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 9(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 9(b)).

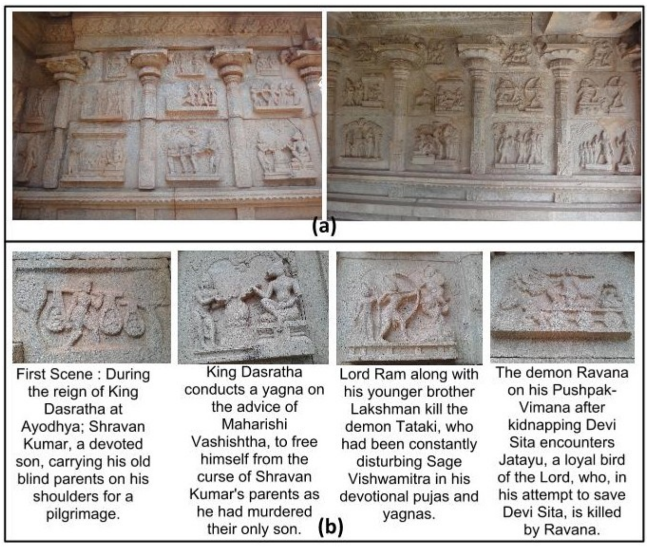
(a)
First Scene : During the reign of King Dasratha at Ayodhya; Shravan Kumar, a devoted son, carrying his old blind parents on his shoulders for a pilgrimage.
King Dasratha conducts a yagna on the advice of Maharishi Vashishtha, to free himself from the curse of Shravan Kumar's parents as he had murdered their only son.
Lord Ram along with his younger brother Lakshman kill the demon Tataki, who had been constantly disturbing Sage Vishwamitra in his devotional pujas and yagnas.
The demon Ravana on his Pushpak-Vimana after kidnapping Devi Sita encounters Jatayu, a loyal bird of the Lord, who, in his attempt to save Devi Sita, is killed by Ravana.
(b)

Figure 9: (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.

6. CONCLUSIONS

Preliminary annotations were collected from local tourist guides. 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. We see this as an initial step to build large-scale image and annotation retrieval applications. Here, we have experimented with a 5K-10K database of images and tried to incorporate a scalable approach that can work with a database of 100K or 1M images. We want to extend our mobile app to successfully retrieve from a larger image database with similar efficiency. Although SIFT is a more robust descriptor, extracting the SIFT descriptors is computationally expensive on a mobile phone and consumes 35-40% of the total time in retrieving annotations. Using faster and compact descriptors could be an interesting alternative. These problems are the future scope of our work.

7. ACKNOWLEDGMENTS

This work was partly supported by Indian Digital Heritage project of DST.

8. REFERENCES

[1] `http://www.snaptell.com/`.
[2] `http://www.kooaba.com/`.
[3] `http://www.pointandfind.nokia.com/`.
[4] `http://www.google.com/mobile/goggles/`.
[5] G. Bernd, V. Chandrasekhar, D. M. Chen, N. man Cheung, R. Grzeszczuk, Y. Reznik, S. Tsai, G. Takacs, and R. Vedantham. Mobile visual search. 2011.
[6] 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.
[7] 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.
[8] 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.
[9] 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.
[10] 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.
[11] 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.
[12] 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.
[13] N. Henze, T. Schinke, and S. Boll. What is that? object recognition from natural features on a mobile phone. In *Workshop on MIRW*, 2009.
[14] 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.
[15] 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.
[16] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.
[17] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.
[18] G. Schroth, R. Huitl, D. Chen, M. Abu-Alqumsan, A. Al-Nuaimi, and E. Steinbach. Mobile visual location recognition. *IEEE SPM*, 2011.
[19] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.
[20] 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.
[21] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *ISMAR*, 2008.
[22] 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.