Optimizing Storage Intensive Vision Applications to Device Capacity

Rohit Girdhar Jayaguru Panda C. V. Jawahar

IIIT-Hyderabad, India

Abstract. Computer vision applications today run on a wide range of mobile devices. Even though these devices are becoming more ubiquitous and general purpose, we continue to see a whole spectrum of processing and storage capabilities within this class. Moreover, even as the processing and storage capacity of devices are increasing, the complexity of vision solutions and the variety of use cases create greater demands on these resources. This requires appropriate adaptation of the mobile vision applications with minimal changes in the algorithm or implementation. In this work, we focus on optimizing the memory usage for storage intensive vision applications.

In this paper, we propose a framework to configure memory requirements of vision applications. We start from a gold standard desktop application, and reduce the the size for a given the memory constraint. We formulate the storage optimization problem as mixed integer programming (MIP) based optimization to select the most relevant subset of data to be retained. For large data sets, we use a greedy approximate solution which is empirically comparable to the optimal MIP solution.

We demonstrate the method in two different use cases: (a) Instance retrieval task where an image of a query object is looked up for instant recognition/annotation, and (b) Augmented reality where computational requirement is minimized by rendering and storing precomputed views. In both the cases, we show that our method allows a reduction in storage by almost $5\times$ with no significant performance loss.

1 Introduction

With the advancement in computing power, data driven methods have gained popularity in solving many challenging computer vision problems. Just as years of visual experience enables humans to make sense out of sparse, noisy and ambiguous local scene measurements, data-driven methods use multiple examples to train the machine (or just remember them) and help in solving the challenging vision problems. Many computer vision techniques use data (eg. images or features from images) for training the vision solution [1, 2]. Often, the data is also required at the testing/inferring stage in these applications (eg. support vectors in kernel SVMs, image patches and dictionaries for inpainting). The situation is more challenging if one uses a template based solution, such as an exemplar SVM [3] or a nearest neighbour scheme [4]. There have been many previous attempts in pruning the templates [5] or support vectors [6] in the pattern

recognition literature. Most recently, Misra *et al.* [7] proposed a technique to compact the set of exemplars used in Exemplar SVM. However, such attempts do not get enough attention as the storage capabilities of desktop machines have increased rapidly in the last few years. This also led to an increased interest in the methods that use large number of images or feature representations at the run time to obtain better quality of results. This resulted in dictionaries or databases of image [8], patches [9], or even feature vectors [10] becoming popular in a wide spectrum of vision applications. In this work, we focus on pruning the storage requirements of such vision applications to suite the mobile device capabilities. *Our focus is not to design a novel mobile vision algorithm.* Rather we start with a vision solution that runs on desktops (considering it as a standard reference), and demonstrate how the memory/storage requirements can be reduced to practically design compact but equally powerful mobile vision applications.

There are many applications that require large visual data at the test stage. For example, Video Google [10] is a object retrieval system designed to find identical instances (images or parts) in large collection of images and videos. This technique has now emerged as the backbone of many product search solutions [11]. Popular and commercial implementations still continue to use a client server implementation where the mobile client is used only for capturing the image and displaying of the product information. A more challenging implementation can compute features and compact representations on the mobile and minimize the communication overheads [12, 13]. In instance retrieval, typically one needs to retain multiple exemplars of the same instance for acceptable performance. The major challenge in such systems is to decide the number of images or feature representations that needs to be maintained in the database. Panda *et al.* [14] demonstrate the instance retrieval on reasonably large dataset (50K) on common mobile phones. However, when the dataset (number of images to be indexed) increases beyond 100K, even this method fails. Focus of [14] was limited to pruning the vocabulary size (or representation) without modifying the dataset of images to be indexed. The problem of automatic selection of relevant exemplar images or their representations still remains as an open problem. A solution to this can result in easy adaptation of the software without changes in the algorithm or even implementation.

Vision on mobile and wearable computers has garnered a lot of interest in the last half a decade. Recent research in this direction is focussing on data-driven and on-device computing approaches. Paucher *et al.* [15] perform indoor localization and pose estimation for AR on mobile devices using a database of images of the environment taken from different locations. As we also show in the experimental section, reducing storage helps in reducing the computational requirement on devices, which have multiple purposes (unlike a dedicated computer). There has been work in optimizing memory and computation of visual search and recognition to work on-device [14, 16, 11]. Computational photography applications on mobile are also focussing on memory optimized design for tasks such as panorama stitching [17]. Pollefeys *et al.* propose novel approaches

overcome the underlying hardware limitations of mobile to accomplish extremely heavy tasks such as live 3D reconstruction [18, 19]. Some systems, however, still choose to offload heavier recognition and detection tasks to servers, while running tracking on the device [20–22].

In this work, we address the problem of developing computer vision applications that can be customized to work on different devices having varying hardware capabilities. We specifically target standalone apps which do all the computations on the device itself, and require large amounts of data to be stored on device. We attempt to reduce the size of such datasets to suit the device capability, without any significant loss in quality of the solution. Consider a standalone, product search application that needs to support more than 100K products, with 10 M exemplars require an index typically of size 40GB in size [23]. Current cloud or server based infrastructure can support this much memory and would be able to answer queries within less than a second. Modern desktops usually support less than 4GB of RAM, but can flatten the dataset to disk, leading to response times of the order 15-35s. High end mobile devices (such as iPhone) support much less disk space (8-16GB), hence can not use the index from disk, and must use an index over a subset. Though 10M examples are collected for 100K products for excellent retrieval performance, we notice that a subset of these images can actually be pruned without any significant loss in performance.

Our main contribution is a fast and simple optimization framework that works for various computer vision problems to select a near-optimal subset of data required for the task to be stored on device. Note that our solution runs on a desktop/server and provides the image/dataset selection information that can lead to an application that fits a specific memory limit. We formulate this selection problem as a mixed integer program (MIP) to select the most relevant subset of data. Integer optimization being NP Hard, the MIP quickly becomes intractable and hence, we use a greedy approximation of the same for larger datasets. We validate our approach in two different vision applications: instance retrieval, and mobile augmented reality for low end devices, by storing and rendering precomputed views.

Even though storage is not a major concern on modern desktops or servers, and in fact leads to more accurate performance, it does become a concern on low-end mobile devices. Moreover, the advent of wearable computing devices in the form of glasses and watches further aggravates the problem, as these typically support even weaker hardware. For example, Google Glass currently support only 682 MB of RAM [24]. Therefore, even though the capabilities of mobile devices have been increasing as a consequence of Moore's law, appearance of computers in miniature formats and growing popularity of data intensive applications still leaves plenty of opportunity in optimizing applications for weaker devices. Even as of today, a significant percentage of mobile devices in the world are incapable of storing or processing large amounts of data to perform complex computer vision tasks. Figure 1 illustrates this fact by giving the percentage market share of android based mobile devices with respect to disk space, RAM and processing power.

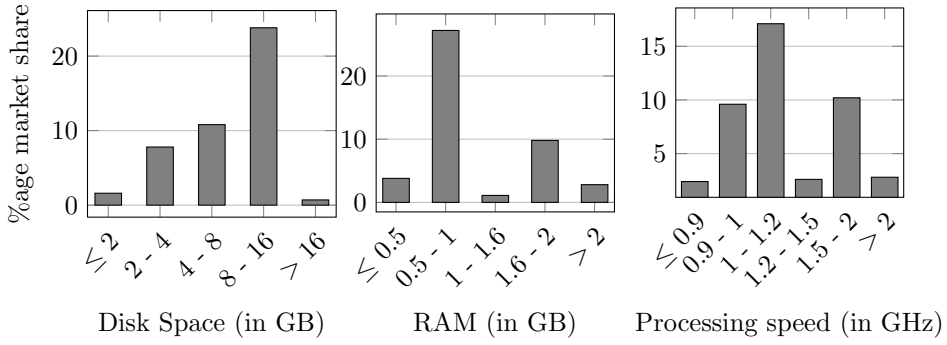

Fig. 1: Market share of android mobile devices by disk space, RAM and processing power in 2014. Even in 2014 we have large number devices with less than 1 GB RAM, and less than 4 GB of disk space. Data from *gsmarena.com* and [25].

We investigate the problem of extracting the most relevant information from a large dataset of visual data. The common theme in most of the work in this direction is to represent the similarity between images in the form of a graph and selecting the nodes that can approximate for the others. One of most relevant works in this direction is by Simon *et al.* on scene summarization [26]. They extract a canonical subset of images as a visual summary of a scene from a large collection of images sourced from the internet. Li *et al.* in [27] further build upon this work to use iconic scene graph with the number of geometrically consistent matches as the edge weights to cluster the images and extract 'iconic' views, which they further use for visualization and search. Crandall *et al.* [28] use a similar approach for organizing large photo collections. A more scalable approach to clustering images at world-scale was described in [29]. Irschara *et al.* [30] use a similar approach to compress 3D scene representation. Chum *et al.* [31] propose similarity measures for detecting near-duplicate images.

In this paper, we propose a simple scalable scheme that can help pruning the dictionaries to suite the mobile devices. We empirically show that pruning can even result in improving the performance in some cases. As an example, we take the instance retrieval and mobile AR problems. In both the cases, we prune dataset of images, with no practical loss in performance. We report a reduction in the number of images to be retained by a factor of 5.

2 Optimizing Memory Usage

2.1 Memory Reduction as Subset Selection

Given a vision solution that uses a database of images $V = \{v_1, v_2, \cdots, v_n\}$ and a given set of possible tasks $T = \{t_1, t_2, \cdots, t_m\}$, we are interested in finding a subset $K \subseteq V$ such that $|K| \leq k$, and can solve the task $t_i, 1 \leq i \leq m$ with minimal impact on accuracy compared to when using the complete set V. For the

case of instance recognition, V corresponds to the set of all images available for indexing, and we are interested automatically selecting in the subset K that gives good performance for the task t_i of recognizing specific products. Performance is measured as precision @ 1 (P_1) for the instance recognition.

Instance recognition and retrieval tasks typically use a lot of examples that are usually sourced from the Internet or collected with minimal supervision. They are also incrementally updated in many situations. Because of the nature of this database and its growth, they tend to contain images that have no role in instance recognition. To verify this, we did a small experiment on the Oxford Buildings dataset [23] consisting of images of famous Oxford landmarks sourced from Flickr. We used a Bag of Words (BoW) representation and observed that almost 33% of the images in this dataset never appear in the top-10 for any of the given queries. In other words, if one uses P_1 as a measure for the recognition accuracy, these images can be pruned from the database with no loss in performance. Figure 2 shows some of them. These images are of two broad types: (i) Images that are redundant, as a better example is sufficient for the recognition. (ii) Images that are really outliers and have not much visual information of interest.

We also observe that the tasks (the possible/popular set of queries) are also not arbitrary. In a typical recognition setting of buildings, users often want recognition of frontal images and not top views. The prior knowledge about the possible tasks t_i can help in computing the loss or cost of a failure due to the pruning of the database. Database can now be designed to support these popular queries and can be compacted further.

Fig. 2: (a) shows the IR results for the left most query, using the full set and a 500 image subset. Note that the top few results in both cases are relevant and sufficient for recognition. (b) shows some of the outlier images in the set.

Hence, we focus on the problem of selecting a k size subset from the dataset V of all elements required for the task, where k depends on the device capability. When the dataset is a simple collection of images, the selection is easy to appreciate. Even when the dataset is computed from a large population (as in the case of a dictionary computed with K-Means [10]), pruning can be used [14].

In our case, the question of *computing* a reduced size dictionary does not really arise since we start with a database of images, and we can only prune it.

2.2 Selecting Useful Images

We now represent the problem as a mixed-integer optimization (MIP) to select a canonical subset (K) of size k that best approximates the complete set (V). The number k would depend on the disk space and memory available on the device. We define e_i as a measure of error incurred when i^{th} image of V is approximated using K. We limit our attention to problems in which a specific image is approximated using only one other image. If the image v_i is an outlier and does not contribute to the performance of the solution, one can trivially remove v_i directly. However, removal of an image that has some utility is more tricky. In this case, the images selected in the attempt to approximate its role can lead to a possibly poorer performance. This quality reduction is represented in e_i.

As discussed earlier, in many situations the tasks are already known, and can be used to drive the optimization to a solution that is optimal for those specific tasks. Let W_{ij} be the weight of image i with respect to task t_j. For example, if task j is product search, W_{ij} will be high for images i that correspond to the most popular views of the product. Now, as our objective is to minimize the total error incurred in approximating set V using set K, for a given task j, it can be represented as:

$$Objective : \min \sum_{i=1}^{|V|} W_{ij} e_i \tag{1}$$

We weigh the error values with W_{ij} to ensure good approximations for more important views. However, this kind of a prior knowledge is usually hard to determine and needs to be inferred from very large set of case studies and user logs. Hence, for the remainder of this problem, we will consider all images to be equally weighted for the task, and take $W_{ij} = 1, \forall i, j$.

Now, for each image in the set V, we define a binary variable x_i, which equals 1 if it is selected to the subset. Let E_{ij} be the error incurred when i^{th} image is approximated using j^{th} image. Since it can not be actually computed, we define it as based on similarity between v_i and v_j. E_{ij} would be task specific, for instance in IR, it could be the distance between GIST descriptors of the images. In case of 3D object augmentation, we can use an image representation over feature tracks to compute the error. Since each feature track corresponds to a 3D point, a low error would ensure same 3D pose is visible in both the images. For even greater accuracy, we could use the distance between extrinsic camera parameters of the two, if such information is available.

In order to control e_i using E_{ij}s, we need to introduce a binary variable Z_{ij} which equals 1 if image i is approximated using image j. Finally, we need to enforce the following constraints: (i) Total number of selected images is limited by k (see Equation 2), (ii) An image can approximate for another image if it itself

gets selected (see Equation 4), (iii) An image may be approximated using only one other image (see Equation 3), and (iv) The error e_i should be determined from E_{ij}, corresponding to the image j that is used to approximate image i (see Equation 5). These translate to the following constraints:

$$\sum_{i=1}^{|V|} x_i \leq k \qquad (2) \qquad\qquad Z_{ij} \leq x_j \qquad \forall i,j \qquad (4)$$

$$\sum_{j=1}^{|V|} Z_{ij} = 1 \quad \forall i \qquad (3) \qquad\qquad e_i \geq \sum_{j=1}^{|V|} E_{ij} Z_{ij} \qquad \forall i \qquad (5)$$

Equation 5 enforces the constraint (iv) as $Z_{ij} = 1$ for a unique value of $j = j'$ for a given i (as an image is approximated by only one other image), and hence, Equation 5 reduces down to $e_i \geq E_{ij'}$, which is what is required by the constraint. The variable x at the minimal value of objective will give the optimal subset of given data set to be selected that would give least approximation error summed over all images. The MIP as described above can be easily transformed into a standard format (such as CPLEX) and solved using existing IP solvers.

2.3 Scaling to Large Sets

Even though MIP is easy to formulate and able to select the optimal subset to store on device, solving itself becomes intractable as the size of the original set increases. We observed that using one of the fastest non-commercial solvers, SCIP [32], the time taken and memory usage for the memory optimization grows exponentially as size of original set ($|V|$) increases. Even though this optimization is an offline task and can be performed using powerful servers, optimizing even over sets of a few hundred thousand images could take few weeks for each device, a timeline typically unacceptable to software application developers.

This motivated us to use a greedy algorithm to get an approximate solution to the above problem. The algorithm starts with an empty K set. At every iteration, we choose the image that leads to maximum reduction in the objective (Q), i.e., the unselected image that best approximates the remaining unselected images, and add it to the selected set. We keep going till k images get selected, and output the final K set. Algorithm 1 formally presents our approach.

Figure 3 compares the performance of MIP optimization vs the Greedy approximate solution. We observed for selecting a subset of $\frac{N}{2}$ from a set of size N, the time taken to optimize MIP (using SCIP) and the memory used in this process increases exponentially. On a desktop with 4GB of RAM, we could not optimize over a set with more than 600 images. The greedy approach, on other hand, was extremely light and fast. The computation time in this case scaled linearly, and we could easily optimize over tens of thousands of images. Moreover, the final objective value at optima in both the cases were comparable for small sets, showing the near-optimality of the greedy approximation. Hence, we use our greedy approach for the further analysis in Section 3, and compare with MIP for smaller subsets where ever possible.

Algorithm 1 Greedy algorithm to select representative subset

$K \leftarrow \emptyset$
while $|K| < k$ **do**
  for all $v \in V - K$ **do**
    $Q_v \leftarrow Q(K) - Q(K \cup v)$
    $Q_{v*} \leftarrow \min(Q_{v*}, Q_v)$
  if $Q_{v*} \leq 0$ **then**
    break
  $K \leftarrow K \cup v*$

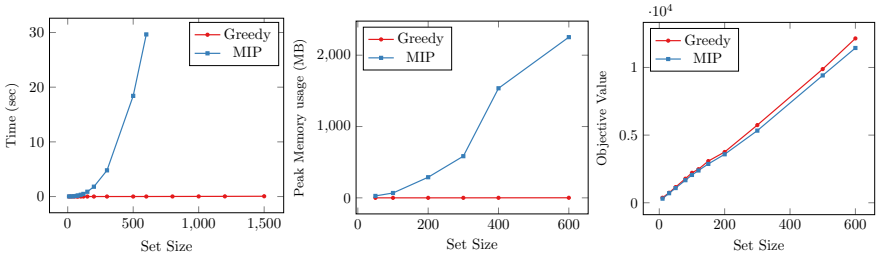

Fig. 3: These plots compare running time, memory usage and objective value for greedy and MIP to select a subset of size $\frac{N}{2}$ from N size sets. Greedy is quite comparable to MIP in performance. MIP, however, takes much more time and memory to compute.

3 Experimental Results

3.1 Example Use Cases

We validate our approach using two popular mobile vision applications, product search using instance retrieval and 3D object augmentation. Instance recognition focusses on recognizing an image related to a specific object given a query from widely varying imaging conditions. Most of the existing IR literature uses variants of bag of words based approaches for the task [33, 23, 34] by matching local image features that represent image geometry to those in database. Image recognition mobile devices also has several successful examples such as Google Goggles, Amazon Snaptell etc, but most of these applications rely on a remote server for matching within large image databases. Some of these optimize on the network communication by sending compressed feature representations to the server [35, 13, 36, 37]. Prior work on offline IR on mobile focusses on reducing memory footprint of search index and includes [14, 38–40]. Though mean average precision (mAP) is used as the measure to quantify the retrieval quality, precision at k (P_k) with $k = 1$ or even 3 or 5 are better suited for the instance recognition task that we are interested in. Since product search or image annotation applications typically use top 1 or 3 matches for recognition, we believe mean precision (mP) to be a better indicator of the performance of our system.

3D object augmentation, on the other hand, involves augmenting natural 3D scenes with virtual objects. The standard approach in such problems is to register the query image with respect to the 3D structure of the scene. The camera parameters thus obtained are then used to transform, render and merge the object onto the query scene. Camera calibration for the query image is the computationally heaviest step in this pipeline, typically achieved by first computing 2D-3D correspondences between the image and 3D structure, and using these to compute camera parameters using RANSAC. As these computations are usually too heavy for a low-end mobile device, we design an approximate solution that can easily be configured to the device capabilities. We pre-compute the augmentation snapshots for a database set of images of the scene. At the test time, the precomputed views of the virtual objects are merged with the input image. If the camera parameters of the precomputed view is very close to that of real one, visually the result is indistinguishable. However, storing thousands of views of a structure might become a bottleneck on lower end devices. Hence, there is a need to prune this database set of views to select best k views to store (where k depends on the amount of disk space available). We use our subset selection technique for the same.

3.2 Experiments and Results

Comparing MIP and Greedy over IR We first compare IR performance over the subsets selected using MIP optimization and the greedy method. We use a bag of words based approach over SIFT vectors quantized into visual words using a 1M vocabulary computed using Approximate K-Means [41]. The scores for ranking are computed using the tf-idf statistic over the visual words, and we further refine the ranklist using spatial consistency constraints, i.e., fitting a fundamental matrix and re-ranking based on number of inliers (we ignore the matches if number of inliers < 15). We evaluate it over a 480 image subset of Oxford Buildings dataset, consisting of the images of All Souls, Ashmolean and Balliol, and their corresponding 15 queries. Number of SIFT inliers is popularly used as a similarity measure between images, and we use its reciprocal to compute the approximation error matrix (E) in both the subset selection approaches. As we can observe from Figure 4, the subsets selected by both MIP optimization and greedy give comparable mP at 3,5 and 10 (though MIP does little better than greedy on average), justifying the usage of greedy for larger sets.

Greedy Subset Selection over Complete Dataset We evaluate the above IR algorithm over the complete Oxford Buildings dataset, consisting of 5062 images of 11 Oxford landmarks with ground truth for 55 queries. We segregate the images into a test set (containing the 55 query images) and a training set (containing the rest 5007 images), and compute the index over the subsets of training set. This helps in evaluating the performance of the system in the real-world product search scenario, where the database does not contain the query images.

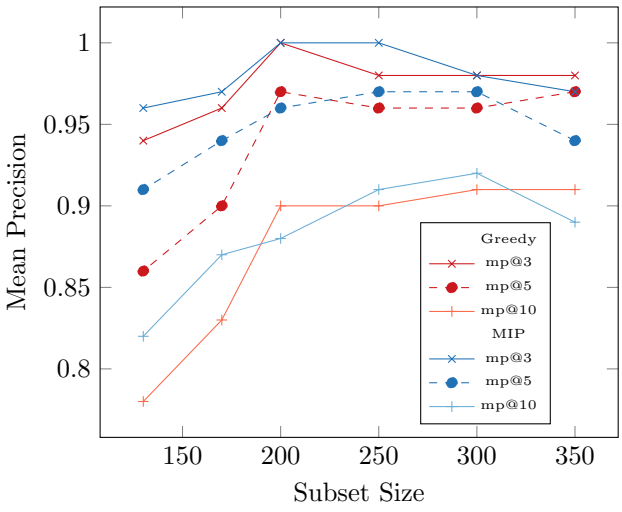

Fig. 4: Graph compares the P_3, P_5, P_{10} values over 480 images subset of Oxford buildings, using both MIP and greedy approaches. Both perform comparably, though MIP is little better on the average.

We experimented with two different values for E_{ij}. GIST [42] is a popular global descriptor for representing scenes, and quite relevant for our problem of removing the similar and outlier images of products. We used the euclidean distance between 512D GIST descriptors as E_{ij}. Another possible way to define similarity between images is as the number of geometrically consistent inlier matches. We also experimented with using reciprocal of this number as the error metric. As we can observe from Figure 5, SIFT inliers are able to model similarity better than GIST, and hence give better mean precision results.

In both cases, we observe from Table 1 that the size of index and computation time goes down almost linearly. This makes our approach suitable for configuring IR applications to mobile devices with all kinds of storage capacities.

Comparing MIP vs Greedy for Augmented Reality In this case, we use a slightly more complex definition of E to ensure images with similar pose incur less error. Hence, we compute SIFT feature tracks across all images, each of which corresponds to a 3D feature point on the structure being augmented (using VisualSfM [43]). Now, we use the images of these feature points to compute the error incurred by using homography to approximate one of these images, by the other. For every 3D point P visible in image i and j, let x and y be the images of P in i and j respectively. Now, E_{ij} is defined as the Euclidean distance between x, and the point obtained by transforming y by the homography between these images, averaged over all such P.

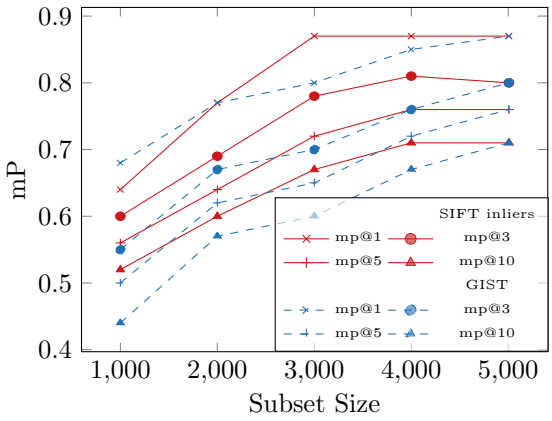

Table 1: Size and speed with subset size

Subset Size (K)	Index Size (MB)	Time (sec)
500	7.33	0.47
1000	12.78	0.76
2000	23.065	1.3
3000	31.221	1.75
4000	38.39	2.17
4500	41.65	2.31
5007	44.62	2.56

Fig. 5: Greedy subset selection results on Oxford Buildings Dataset. We compare the mean precision (mP) at positions 1,3,5 and 10 for GIST and SIFT inliers as the similarity measures between images. In both the cases, size of index stored on device and time taken for search comes down almost linearly.

For experimentation and analysis, we created our own dataset of a toy 3D object in the lab setting. We collected 96 images of the object from different viewpoints, and hand-picked 15 images as the test set. We used a 3D model of a cap to augment the structure, and precomputed the augmentation snapshots of the cap for all the database views. Figure 6 gives a visualization of the same.

We used the greedy and MIP optimization strategies of Section 2 to select subset from the remaining 81 images. We quantitatively evaluate the performance using mean reconstruction error over test images, defined in a similar way as E. Reconstruction error for image i is defined as

$$\min_{j \in K} E_{ij}$$

and is averaged over all test images to get the mean (E_{ij} is defined in Section 2.2). We compute the mean reconstruction error for the 15 test images and plot it

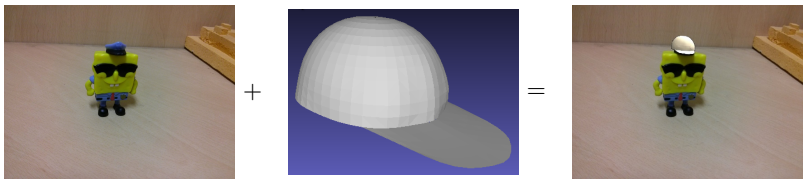

Fig. 6: An image of the toy 3D model from our dataset, and a snapshot of the cap used for augmentation. A snapshot of the cap corresponding to the first image is used to generate the final result

against the subset sizes in Figure 7. Interestingly, the greedily selected subset at times gives even lesser error than the MIP optimized subset. This is because MIP tends to overfit the solution to the training data, giving the most optimal subset that can reconstruct the other training images, with no consideration to test images. Greedy, on the other hand, does not have such tight constraints and hence tends to generalize better. We also show qualitative results of augmentation with different subset sizes in Figure 7.

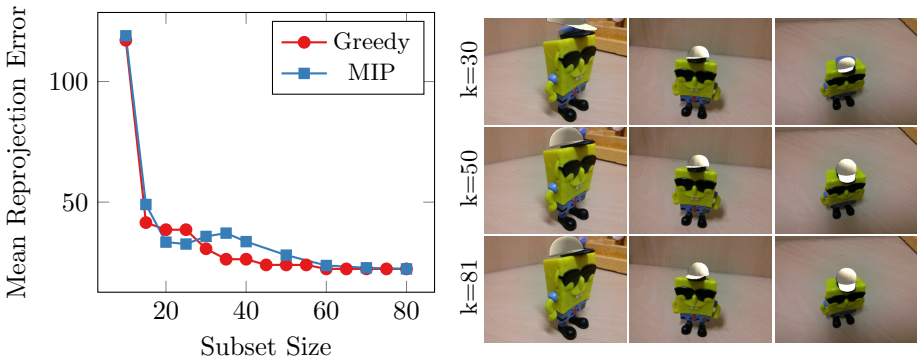

Fig. 7: The graph shows the variation of reprojection error as we use smaller sets of precomputed views. We also show augmentation results for 3 different views of our 3D object at different values of subset size k. Clearly, reducing the size of selected subset has marginal effect on quality of augmentation.

3.3 Applications in Digital Heritage

Recently, there has been a rising interest in building computer vision applications for digitally preserving and promoting cultural heritage, such as the The Great Buddha Project [44], HeritageApp [45] etc. The 3D augmentation system too has a natural application in digital heritage, especially in augmenting 3D structures with parts that no longer exist. Stone Chariot at Hampi is a famous historical monument in India (Figure 8). Its structure went through various changes over the course of history; the most prominent one being the removal of a dome like super-structure from top of the monument. With the objective of preserving this cultural heritage, a 3D model was built that captures the structure in its former glory (Figure 9) We aim to make this visualization accessible to people on their mobile devices. Our application allows any tourist to see the missing parts augmented over the original structure, using any mobile device. Some of the results of our homography based approach are shown in Figure 10.

This dataset consists of 1505 images of a heritage monument taken from different viewpoints and the precomputed augmentation snapshots occupy nearly

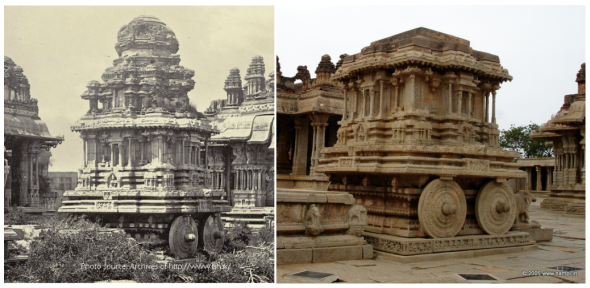

Fig. 8: The Heritage structure in 1856 (courtesy *hampi.in*) and now. Note the dome-like superstructure that no longer exists.

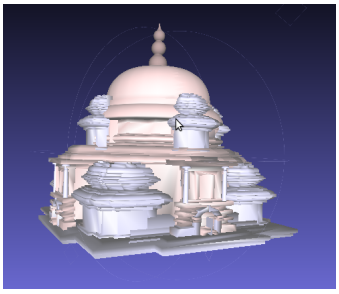

Fig. 9: The 3D model of the missing dome constructed by architects.

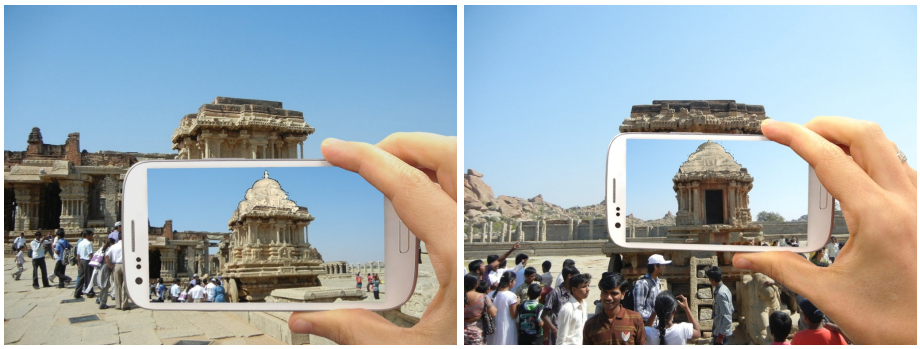

Fig. 10: Example use of our application to augment the missing parts.

420 MB on disk. We reduce this requirement using the greedy subset selection to enable it to run on devices with low disk space. Figure 11 shows the marginal increase in reprojection error with reduction in size of subset using the greedy approach. Table 2 lists the various specifications of the app for different subset sizes for a Lenovo s820 mobile device. These include the disk space occupied by precomputed snapshots, size of the search index used for localization, startup time of the app to read data into memory, time in instance retrieval over database views for localization and finally the time to compute matching, homography and warping and merging the precomputed snapshot onto the query. As expected, total computation time and memory load decreases with decrease in the subset size.

3.4 Discussions

Apart from low processing power, less memory capacity, small screen size and limited connectivity, another major challenge faced by mobile devices is limited battery power. Even as the memory and processing power continue to grow exponentially according to the Moore's law, growth in battery technology has

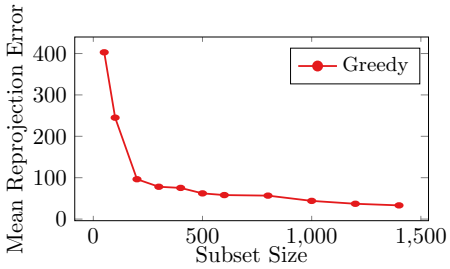

Table 2: AR Mobile app specs

Subset size	**500**	**1000**	**1505**
Snapshots (MB)	140	278	422
Search Index (MB)	7.3	12.7	18.6
Startup Time (sec)	0.7	1.2	1.6
Localization (sec)	0.03	0.07	0.1
Augmentation (sec)	0.6	0.6	0.6

Fig. 11: The graph of mean reprojection error with subset size shows for the 1505 image heritage monument dataset. The table lists our app specifications for different subset sizes.

been much slower. For instance, while Apple claims that the iPhone 5S delivers 56× faster graphics and 40× faster CPU than original iPhone [46], the battery capacity of 5S has grown by only 15% from the original [47].

Clearly, such a situation warrants the need for more battery conscious applications. Fortunately, our configurable application approach fits this scenario well, too. Since using smaller subset of visual data requires lesser computation and hence lesser battery consumption, a smart application can automatically shift to using a smaller subset as the device goes low on power. The different subsets may either be stored on device or can be retrieved on the fly over the network. Once the device charges up again, the application can again shift to using complete data for more accurate performance.

4 Conclusion

In this work, we successfully formulate a fast and simple approach for pruning datasets required by storage intensive applications, enabling them to work across a spectrum of devices with varying capabilities. Even though finding the most optimal such subset is computationally infeasible, we show that a greedy approach can closely approximate that performance. We validate this approach over popular vision applications, IR and Augmented Reality, showing significant reductions in storage and computation requirements while nearly preserving the accuracy of performance. We also demonstrate its application in digital heritage, by enabling low end mobile devices to visualize the parts of heritage monuments that were lost over time.

Acknowledgement. This work was partly supported by the Indian Digital Heritage Project of the DST.

References

1. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR. (2005)

2. Judd, T., Ehinger, K., Durand, F., Torralba, A.: Learning to predict where humans look. In: ICCV. (2009)
3. Malisiewicz, T., Gupta, A., Efros, A.A.: Ensemble of exemplar-svms for object detection and beyond. In: ICCV. (2011)
4. Zhang, H., Berg, A., Maire, M., Malik, J.: Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In: CVPR. (2006)
5. Fayed, H.A., Atiya, A.F.: A novel template reduction approach for the k-nearest neighbor method. IEEE Transactions on Neural Networks (2009)
6. Liang, X.: An effective method of pruning support vector machine classifiers. IEEE Transactions on Neural Networks (2010)
7. Misra, I., Shrivastava, A., Hebert, M.: Data-driven exemplar model selection. In: Winter Conference on Applications of Computer Vision. (2014)
8. Hays, J., Efros, A.A.: Scene completion using millions of photographs. In: ACM SIGGRAPH. (2007)
9. Peyr, G.: Sparse modeling of textures. Journal of Mathematical Imaging and Vision (2009)
10. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: ICCV. (2003)
11. Xiaohui Shen, Zhe Lin, J.B., Wu, Y.: Mobile product image search by automatic query object extraction. In: ECCV. (2012)
12. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: An efficient alternative to sift or surf. In: ICCV. (2011)
13. Chandrasekhar, V., Takacs, G., Chen, D.M., Tsai, S.S., Reznik, Y., Grzeszczuk, R., Girod, B.: Compressed histogram of gradients: A low-bitrate descriptor. IJCV (2012)
14. Panda, J., Brown, M.S., Jawahar, C.: Offline mobile instance retrieval with a small memory footprint. In: ICCV. (2013)
15. Paucher, R., Turk, M.: Location-based augmented reality on mobile phones. In: CVPR Workshop. (2010)
16. He, J., Feng, J., Liu, X., Cheng, T., Lin, T.H., Chung, H., Chang, S.F.: Mobile product search with bag of hash bits and boundary reranking. In: CVPR. (2012)
17. Xiong, Y., Pulli, K.: Fast image stitching and editing for panorama painting on mobile phones. In: CVPR Workshop. (2010)
18. Kalin Kolev, Petri Tanskanen, P.S., Pollefeys, M.: Turning mobile phones into 3d scanners. In: CVPR. (2014)
19. Tanskanen, P., Kolev, K., Meier, L., Camposeco, F., Saurer, O., Pollefeys, M.: Live metric 3d reconstruction on mobile phones. In: ICCV. (2013)
20. Dantone, M., Bossard, L., Quack, T., Van Gool, L.: Augmented faces. In: ICCV Workshops. (2011)
21. Gammeter, S., Gassmann, A., Bossard, L., Quack, T., Van Gool, L.: Server-side object recognition and client-side object tracking for mobile augmented reality. In: CVPR Workshop. (2010)
22. Shyam Sunder, K.M.S., Savarese, S.: Mobile object detection through client-server based vote transfer. In: CVPR. (2012)
23. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR. (2007)
24. : Google glass. (en.wikipedia.org/wiki/Google_Glass) Accessed : June 19, 2014.
25. : Top android phones. (http://www.appbrain.com/stats/top-android-phones) Accessed: 2014-06-03.
26. Simon, I., Snavely, N., Seitz, S.M.: Scene summarization for online image collections. In: ICCV. (2007)

27. Li, X., Wu, C., Zach, C., Lazebnik, S., Frahm, J.M.: Modeling and recognition of landmark image collections using iconic scene graphs. In: ECCV. (2008)
28. Crandall, D.J., Backstrom, L., Huttenlocher, D., Kleinberg, J.: Mapping the world's photos. In: WWW. (2009)
29. Quack, T., Leibe, B., Van Gool, L.: World-scale mining of objects and events from community photo collections. In: International Conference on Content-based Image and Video Retrieval. (2008)
30. Irschara, A., Zach, C., Frahm, J.M., Bischof, H.: From structure-from-motion point clouds to fast location recognition. In: CVPR. (2009)
31. Chum, O., Philbin, J., Zisserman, A.: Near duplicate image detection: min-hash and tf-idf weighting. In: BMVC. (2008)
32. Achterberg, T.: Scip: Solving constraint integer programs. Mathematical Programming Computation (2009)
33. Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: CVPR. (2006)
34. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: ICCV. (2003)
35. Girod, B., Chandrasekhar, V., Chen, D.M., Cheung, N.M., Grzeszczuk, R., Reznik, Y., Takacs, G., Tsai, S.S., Vedantham, R.: Mobile visual search. SPM (2011)
36. Chandrasekhar, V., Reznik, Y., Takacs, G., Chen, D., Tsai, S., Grzeszczuk, R., Girod, B.: Quantization schemes for low bitrate compressed histogram of gradients descriptors. In: CVPR. (2010)
37. Chen, D.M., Tsai, S.S., Chandrasekhar, V., Takacs, G., Singh, J., Girod, B.: Tree histogram coding for mobile image matching. In: DCC. (2009)
38. Jégou, H., Douze, M., Schmid, C.: Packing bag-of-features. In: ICCV. (2009)
39. Jégou, H., Perronnin, F., Douze, M., Sánchez, J., Pérez, P., Schmid, C.: Aggregating local image descriptors into compact codes. TPAMI (2012)
40. Zhang, X., Li, Z., Zhang, L., Ma, W.Y., Shum, H.Y.: Efficient indexing for large scale visual search. In: ICCV. (2009)
41. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR. (2007)
42. Oliva, A., Torralba, A.: Modeling the shape of the scene: A holistic representation of the spatial envelope. IJCV (2001)
43. Wu, C.: Visualsfm: A visual structure from motion system (2011)
44. Ikeuchi, K., Oishi, T., Takamatsu, J., Sagawa, R., Nakazawa, A., Kurazume, R., Nishino, K., Kamakura, M., Okamoto, Y.: The great buddha project: Digitally archiving, restoring, and analyzing cultural heritage objects. International Journal of Computer Vision (2007)
45. Panda, J., Sharma, S., Jawahar, C.V.: Heritage app: Annotating images on mobile phones. ICVGIP (2012)
46. Heath, A.: iphone 5s is 56x faster than original iphone with 64-bit a7 chip. (http://www.cultofmac.com/244572/iphone-5s-is-56x-faster-than-original-iphone-with-64-bit-a7-chip/) Accessed: June 5, 2014.
47. Drang, D.: The small improvement in iphone battery capacity. (http://www.leancrew.com/all-this/2013/10/the-small-improvement-in-iphone-battery-capacity/) Accessed: June 5, 2014.