# TECHNIQUES FOR ORGANIZATION AND VISUALIZATION OF COMMUNITY PHOTO COLLECTIONS

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

*Master of Science*
*in*
*Computer Science*

by

Kumar Srijan
200602015
kumar.srijan@research.iiit.ac.in

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

International Institute of Information Technology
Hyderabad, India

# CERTIFICATE

It is certified that the work contained in this thesis, titled " TECHNIQUES FOR ORGANIZATION AND VISUALIZATION OF COMMUNITY PHOTO COLLECTIONS " by **Mr. Kumar Srijan**, has been carried out under my supervision, and is not submitted elsewhere for a degree.

Date

Adviser: Prof. C.V. Jawahar

To the Blue Planet

# Acknowledgments

I would like to thank my advisor Dr. C. V. Jawahar for giving me an opportunity to do research at CVIT, IIIT Hyderabad. I am indebted for his support and guidance. I would also like to thank Dr. P J Narayan, Dr. Jayanthi Siwaswamy and Dr. Anoop Namboodri for various references and guidance in different subjects related to the stream. I wish to express my gratitude to Mr. Pratuysh Kumar Singh for invaluable lessons in maths and physics.

I am also grateful to fellow lab mates at the CVIT, IIIT Hyderabad for their stimulating company. I am very thankful to Siddharth Choudhary, Atif Iqbal, Vaibhav Kedia, Raman Jain, Vempati Sreekanth, Vibahav Vineet, Pawan Harish, Pramod Sankar, Omkar Parkhi, Anand Mishra and Gopal Joshi for their support and guidance during these years. I would like thank my friends and well wishers who always insisted me to work hard and always believed in me. Above all, I am thankful to my family members for their patience, support and love.

# Abstract

Due to the digital and information revolution we are witnessing presently, there are a huge and continously increasing number of images present on the Internet. For example, a query for "Eiffel Tower" on Google Images returns more than two million images. The easy accessiblity of this data provides us with unique opportunities to mine the contents of these images not only to do automatic organization, but also for providing interactive interfaces to browse, explore and query. This task is challenging given the massive size and the continous growth of the collection. To add to this, these collections are taken in varying imaging conditions, with different cameras, at different resolutions, from different perspectives and have different degrees of occlusions present in them. Hence, for image collections even the simplest of tasks such as finding matching images turn out to be hard.

The Computer Vision community has been actively designing and redesigning algorithms to overcome these challenges. One of the most widespread and noticable idea employed is that of extracting robust, invariant and repeatable local features in the images, followed by the subsequent quantization of the feature space as visual words. The similarity of images is gauged by the correspondence and similarity of thier local features. Verifications of the matchings is done to eliminate spurious matches. Building a data structure such as an inverted index over these visual words can catalyse the process of discovery of matching features. This mining of similar images by matching features, forms the basis of all high level algorithms such as clustering, skeletonization, summarization etc. which help in the organization, exploration and querying of these image collections. This thesis presents two novel algorithms which help in achieving this goal.

First, we introduce a novel indexing scheme that makes it possible to do exhaustive pairwise matching in large image collections. The quantization of image features and thier indexing provide on a limited amount of leverage for speeding up the image matching process which depends upon the sparsity the posting lists. This sparsity is controlled by the number of visual words used which after a point cannot be increased arbitrarily without affecting recall. Our scheme, generates higher order features by pairing up nearby features and encoding their affine geometry. This provides a much larger feature space to index which can be subseqently reprojected to any desired size by defining appropriate hash functions. We implement our indexing scheme by providing an analogy with Bloom filters. The higher order features extracted in the images are inserted into their respective equally sized Bloom filters using a single hash function. This unformity in Bloom filters allows for only a single inverted index to be able to index the hash buckets of all the Bloom filters, and thus providing a simplified interface to implicity

query all the Bloom filters. We choose the size of these Bloom filters to be in proportion to the size of the database. This enables us to do querying in constant time, since the average size of the posting lists becomes constant. Also, the use of such large implicit Bloom filters is able to sufficiently mitigate the negative effects of using a single hash functions. As a result, we are able to do exhaustive pairwise matching over large databases of upto 100K images in linear time complexity.

Second, we present a fast and easy to implement framework for browsing large image collections of landmarks and monumental sites. The existing framework "Phototourism" would require doing a reconstruction of the whole scene by employing Structure from Motion package called Bundler. This requires pairwise matching required to generate tracks of matching features across images. Next an incremental approach is applied, starting with a seed reconstruction and adding more matching images into the reconstruction. This, however, requires continous refinement of the whole reconstruction using a computationally expensive procedure called bundle adjustment. The pairwise matching and bundle adjustment become the limiting factors in scaling this technique to large image collections.

To overcome the issues faced with "Phototourism", our framework employs independent partial reconstructions of the scene. We use standard Bag of words model and indexing techniques to determine closest neighbours of each image in the collection, and do a local reconstruction corresponding to each image using only the neighbouring images. This requires us to only solve multiple simple reconstructions problems instead of one large reconstruction problem, making it computationally more tractable. Our browsing interface hops from one reconstruction to another to give the user an illusion of browsing a global reconstruction. Our approach also makes it easy to adapt to growing image collections, as adding an image only incurs a cost of creating a new independent reconstruction. We validate our approach with a Golkonda Fort image dataset consisting of 6K images.

In summary, the techniques presented in this thesis for organizing large image collections tries to solve the problem of doing exhausitive pairwise matching in image collection in a scalable manner, for which a novel indexing scheme is proposed. We also present a novel technique for overcoming the problems faced while doing "Structure from Motion" on large image collections. We hope that these techniques will find application for browsing and mining matching images in large image collections, and also in creating virtual experiences of several monuments and sites across the globe.

# Contents

# List of Figures

# List of Tables

*Chapter 1*

# Introduction

When ordinary light falls upon objects, it gets dispersed in all directions, conveying the information about the surface of the objects. We have a sense of sight which enables us to see objects, both far or near, as our eyes are able to gather these light rays, and our brain is able to make sense of this input. Throughout the history, we have always been interested in making visual records of events happening around us. Many ancient civilizations have made carvings on stones resembling humans, animals, heavenly bodies etc. The fidelity of this method was superseded by paintings as it introduced color, and allowed a higher degree of artistic expression. In this regard, we are currently living in the age of digital photography. A digital camera is a device which essentially converges and captures light rays to produce a permanent impression, called an image or a photograph. An image, therefore, acts as a high fidelity record of what the objects in the world would look like while standing at a place and looking in some direction. Images are, therefore, an invaluable means of making visual records of objects and events happening around us.

In this thesis, we are interested in images of monuments, landmarks, historical sites and buildings. We will look in the direction of organizing and visualizing such image collections which are now easily accessible through the internet. This will help in providing an interactive platform where people could virtually experience visiting a site.

## 1.1 Community Photo Collections

An image vividly describes a part of a moment in the real world. Their effectiveness, however, is limited not only by the amount of space they capture and the level of detail, but also by the fact that they capture information about a single instance in time. For instance, a user may be interested in knowing what is to the left, what is to the top, what would the scene look like if I step back or zoom in. In such a case, other images of the same context can come to rescue.

Gone are the days when we had to wait for the photographs to get developed before we can see the image. With digital cameras, we can almost instantly see the image captured. This is made possible by the use of CCD sensors which can convert the light signal into digital signals. This allows the image to

be compressed and stored in popular formats such as jpeg, png etc. Now, a large number of photographs can be acquired by a single person in one go. With the advent of high speed internet connectivity, many people share these photo collections on the Internet, making them accessible to everyone. Therefore, now it is possible for anyone to acquire and share a large number of photographs of a given site. These collections are called *Community Photo Collections*. These collections are very interesting as it is possible to find varied related images for a given image to enhance its content. Specifically, for heritage sights and tourist destinations, these collections can be used for providing a virtual sense of being at the site. In this thesis, we will present techniques to organize and visualize such image collections.

### 1.1.1 Harvesting Community Photo Collections

Various photo sharing sites, such as Flickr, Panoramio etc. allow the users to annotate their images with tags. On the other hand, users can do a tag based search to retrieve these images. A query about a famous site such as "Taj Mahal" on Flickr can return up to 250k results. Downloading these images would be a cumbersome task, but fortunately, these sites provide support for automatically download these images. Similarly, web services such as Google Images crawl the web and index the images thus found. Users can search in these images based on keywords. For example, a keyword search for "Taj Mahal" on Google Images can returns up to 21M images. Figure 1.1 shows a sampling of images obtained by searching for a query "Golkonda Fort" on Google Images.

### 1.1.2 Challenges in Organizing the Community Photo Collections

Given the crowd sourcing model of Community Photo Collections, they tend to be very diverse. Images having a particular tag may be acquired with cameras ranging from DSLRs to Mobile Phone cameras, they capture the site under various illumination conditions and from various viewpoints. These images also differ very much in their resolution, and also on the amount of metadata embedded into them. Some of the tags may be inaccurate, and such images need to be filtered. More than 80M images get uploaded to the web everyday. As a side effect of this, the community photo collections are also growing at a very rapid rate. These characteristics make these collections an interesting resource for visualizing the landmarks, but at same time pose serious challenges as well. For example, given the large variations in lighting, occlusion and viewpoint, it is not possible to reliably identify similar images by using direct matching methods like cross correlation.

## 1.2 Structure from Motion

Structure from Motion(SfM) refers to the process of computing the three-dimensional structure using the motion information available in images or videos captured from different view points. Traditionally, 3D points are computed given the camera poses of the input images, using a process known as *triangulation*. Conversely, if we have known 3D point coordinates, the camera pose of the given image can

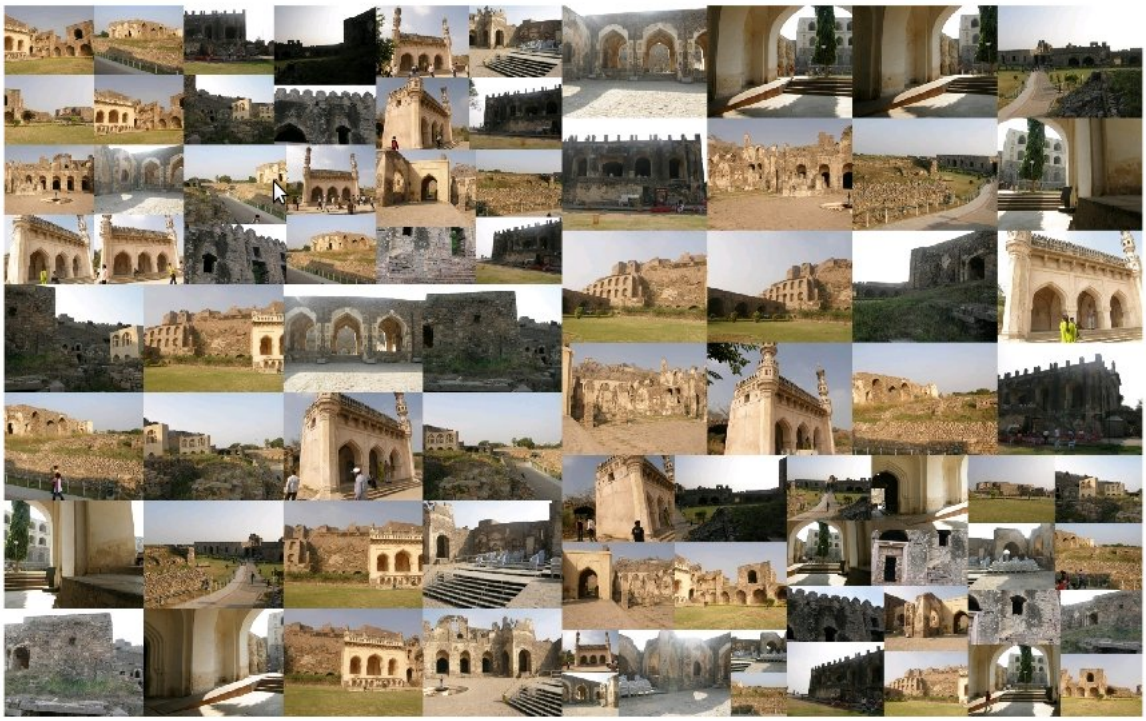


**Figure 1.1** A sampling of images obtained by searching for “Golkonda Fort” on Flickr.

be estimated using pose estimation. However, in structure from motion neither camera pose nor point location is available. This is an example of a circularly related problem. SfM estimates the camera poses and the scene structure simultaneously without requiring either to be known a priori, from the information about common points visible in the images.

### 1.2.1 Multi View Stereo

In this setting, several cameras with known internal and external calibrations are used to image an object. This helps in recovering a dense mesh of the shape of the object, as each region on the surface is imaged by several cameras. Figure 1.2 shows a dense reconstruction of a toy dinosaur recovered from an image sequence. The famous fight sequence in the movie “The Matrix” was captured by using this technique. Figure 1.3 shows a few snapshots of the sequence.

### 1.2.2 Two-View Structure from Motion

Two view SfM is closely related to the depth perception in the binocular human vision. Using images formed by an object in two eyes, we are able to roughly triangulate its distance in the real world. To do this, our brain implicitly computes the correspondences between the two images, so that we know which

(a) (b)

**Figure 1.2** Multi View Stereo: (a) A sampling of toy dinosaur images (b) A 3D reconstruction obtained by applying a multi view stereo algorithm. Image credits : Yasutaka Furukawa

**Figure 1.3** An action sequence in the movie "The Matrix" made possible by the use of multi view stereo.

point in the two images correspond to the same 3D location. Our brain uses the distance between the two eyes (similar to relative camera pose) to triangulate the 3D location of the object. The same technique can be applied to solve the problem of two-view SfM. Given two images of the same location, we find matching pixels in the two images and use those matches to estimate the pose of one image relative to the other. Using the relative pose, we estimate the 3D location of a point corresponding to each pair of matching pixels. To do so, we shoot a 3D ray from each camera location through their respective matched pixels and intersect the rays to get a 3D point location. Figure 1.4 depicts the triangulation of a point using two views. The basic algorithm to estimate 3D point for two-views consists of these three steps,

- **Correspondence Computation** : Identify corresponding points in the two images.
- **Triangulation** : Project rays from the corresponding points; their intersection gives the 3D position of the point.
- **Pose estimation** : Estimate the relative camera pose between the two views.

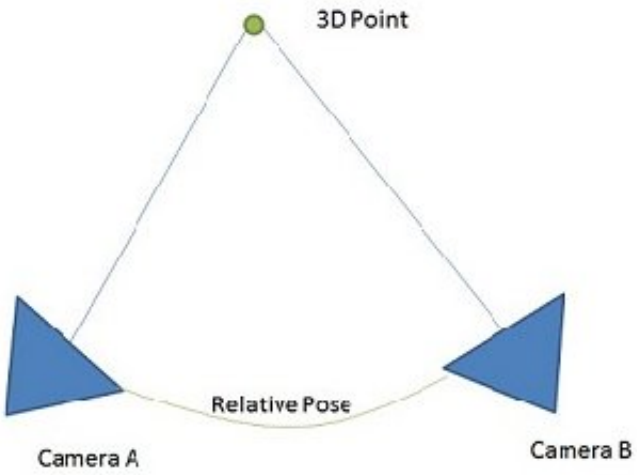



**Figure 1.4** Two-view Structure from Motion

### 1.2.3 Structure from Motion on Community Photo Collections

Structure from motion on Community Photo Collections is a more difficult problem than Multi View Stereo simply because the input to the former are "images taken in the wild" about whom very little meta information is known. In contrast, while doing Multi View Stereo images are taken in a controlled settings. Recently there has been a growing interest in calibrating cameras using the information available in the images itself without using any external aid in calibration. In the case of two-view SfM, pose of an image relative to the other image is estimated by finding 2D-2D correspondences between the two images. Given the correspondences, five-point algorithm is used to figure out the relative camera pose of two views [22, 36]. The 2D-2D correspondences are triangulated to estimate 3D points. Similar

**Figure 1.5** Incremental reconstruction of " Trevi Fountain" by incremental addition of images in Photo-Tourism [52]. Image reproduced from [50].

technique can be extended for community photo collections where multiple uncalibrated images are present.

Large scale sparse 3D reconstruction from community photo collections using the structure from motion pipeline is an active research area today. The process poses many algorithmic challenges due to large variations in the weather, illumination conditions, camera type and the lack of geographic information. Figure 1.6 gives a flowchart of the whole pipeline for reconstructing multiple images by structure from motion as used in Photo-tourism [52]. The two major steps involved are : (a) Correspondence estimation via feature matching and (b) Incremental structure from motion, which we briefly review next.

#### 1.2.3.1 Correspondence Estimation via Feature Matching

In order to construct 3D points corresponding to 2D points on the images, local features have to be computed in each image which are matched between every two image. In order to take into account the large variations, SIFT keypoint detector and descriptor [26] is used to find features in each image. SIFT descriptors are invariant to scale changes and are quite robust to other variations. Therefore, it is well suited to find correspondence across images. SIFT ratio test based matching is used to find correspondence between two images [26]. The pair-wise image matches are geometrically verified using epipolar constraints into RANSAC [12]. It removes the outliers which are not geometrically consistent and gives a relative pose within all pair-wise configurations. Once pairwise matching and geometric verification is done, all the matching features corresponding to the same 3D point are organized into a track. These tracks serve as the basis of doing reconstruction, since it represents various views of a single point in the real world.

### 1.2.4 Incremental Structure from Motion

Initially, we find the best matching pair of cameras which are optimally separated and reconstruct it using the two-frame reconstruction method. Given the *tracks* of matching features computed earlier, we estimate the 3D point and camera pose using Structure from Motion. Rest of the images are added incrementally in small batches and reconstructed. After every increment, the points and cameras are optimized using a non-linear optimization called *bundle adjustment* which minimizes the sum of re-projection error across all registered images. The final output is a sparse point cloud corresponding to tracks and the determination of position and orientation of the cameras which took the photos. Figure 1.5 visualizes the result of reconstruction after every set of images are added.

## 1.3 Problem Setting

We identify the following two problems as being very impactful for organization and visualization of Community Photo Collections.

- Automatic discovery of matching images.
- Overcoming the scalability bottlenecks faced by incremental Structure from Motion pipeline.

## 1.4 Contributions

In this thesis we have made two contributions relating to the organization of Community Photo Collections:

- Design an algorithm for doing exhaustive pairwise matching in large image collections to create image match graphs.
- Design an efficient system for creating walkthroughs in large image collections while also allowing incremental addition of new images.

## 1.5 Organization of the Thesis

- Chapter 2 provides a general introduction to the theory and concepts relevant to this thesis.
- Chapter 3 talks surveys a popular indexing technique used for organizing image collection. We use Bloom Filters for efficient representation of the appearance and geometry of the features extracted in an image data. We also introduce a new indexing scheme built on top of Bloom filters for doing efficient retrieval.
- Chapter 4 investigates the terms required to make pairwise image matching feasible in order to build match graphs on large image collections. We use the tools and techniques developed in Chapter 3 to do pairwise matching and subsequently build a match graph on 105K images in linear time complexity.
- Chapter 5 talks about the problem of browsing large image collections. We survey a popular technique called *photo-tourism* to find its limitations, and provide our solution by creating scene walkthroughs using independent local reconstructions.
- Chapter 6 concludes the work with a discussion and scope for future directions of research.

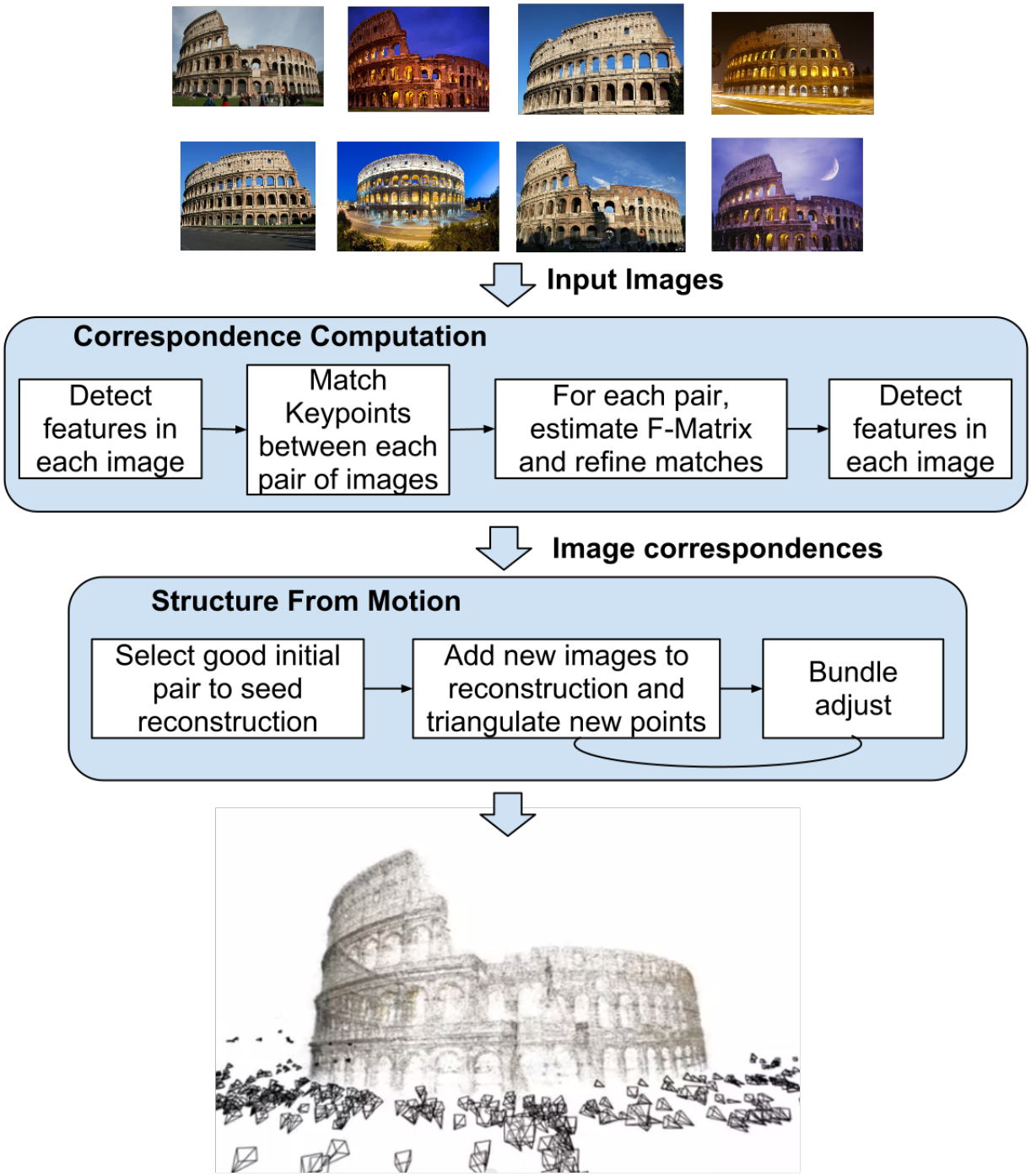



**Figure 1.6** Pipeline for Incremental Structure from Motion used in Photo-Tourism [52].

*Chapter 2*

# Background

## 2.1 Image Matching

Image matching is the most fundamental step towards automatic processing and organization of image collections. The performance of various computer vision tasks such as object recognition, image retrieval, image mosaicing, 3D reconstructions, etc. depends on the effectiveness of the underlying matching scheme. The notion of matching images is also different for different applications. For example, in "Object Retrieval", an image of Ram's car would be considered different from Shyam's car; from the point of "Categorization" or "Object Class Retrieval" problem, these two cars are considered matching as they refer to the same concept – car.

Image are matched based on the features extracted from them. These features act as a representation scheme which allows us design a measurement of similarity. A good feature representation scheme should have the following properties:

- Efficient to compute
- Discriminative representation
- Invariance to distortions or variations
- Robustness to noise

### 2.1.1 Image Representation

An image is stored as a grid of color intensity. A cell in the grid is called a pixel. Black and white images have only one channel which encodes the intensity of whiteness at each pixel location. For a colored image we have three channels for the three primary colors, viz. red, green and blue. All other colors are approximated by various combinations of intensities of these colors. One may note that it is difficult to match images at this level.

### 2.1.2 Feature extraction

The content of matching images can be very different. For example, two images can be showing the same car, but at different locations. The corresponding pixel in the two normalized images may not correspond the same portion of the car. Therefore modelling the image matching problem as a correspondence finding problem is widely employed. In the current context, correspondence computation can be simply understood as which regions of one image match to other regions in another image. In this case, it is beneficial to extract features corresponding to local regions in the image. As a first step towards this, interesting regions in the image are found, and subsequently discriminative descriptions of the regions are computed. In this thesis, we shall be using the Affine-covariant interest regions and SIFT [26] interest point detector and descriptor which we review next.

#### 2.1.2.1 Affine-covariant Interest Regions

Local feature based approaches select patches from the image to obtain a cumulative representation of the image. For higher efficiency, one should identify and use patches that are bound to be invariant under any possible transformations due to various distortions. Local feature based approaches can represent and reliably recognize a variety of real-world images. One strict requisite is that the images should have at least light textures. The key to good performance in this setting are (a) repeatability of the local regions across various instances of a class (b) robust and discriminative description of local regions (c) redundant description of an image by the use of multiple local regions to cope with occasional missed or mismatched regions.

To achieve the above stated objectives the first problem is to use a consistent region detector to mark patches for representation in the image. A popular class of region detectors find a set of Affine covariant regions [31] in the image. The basic idea behind these regions is that the shape of the region is automatically adapted to underlying image intensities in a single image in such a way that regions detected independently in each image correspond to the same 3D surface patch. Other region detectors such as [25] choose regions that are PCA or extrema of the DoG operator response. Patch orientation in those approaches is chosen based on local image gradient orientations. The innovation in affine covariant regions lies in automatically determining the shape of the local region. Regions described in this section are called affine covariant because their size and shape transforms covariantly with a 2D affine image transformation. An affine transformation is a reasonable good local approximation to transformations arising from viewpoint changes for locally planar(or at least smooth) surfaces.

Several affine covariant region detectors have been proposed in the literature [33], examples include

- Iterative region shape adaptation about an interest point [30].
- Selecting stable areas from an intensity watershed segmentation [27].
- Parallelogram growing starting from an interest point [60].

- Exploring intensity profiles along lines emanating from a local intensity extrema [60].
- Searching over elliptical region support for extrema of a saliency measure [20].

A comprehensive review of affine covariant region detectors, and a comparison of their performance appears in Mikolajzcyk et al. [33].

#### 2.1.2.2 Feature Descriptors

After detecting local regions in an image, the usual procedure is to extract a descriptor from each region, which is then used for image matching. Some methods for extracting local image descriptors are reviewed next. A fairly comprehensive review of local region descriptors and experimental comparison of their performance is given by Mikolajczyk and Schmid [32]. Some traditional feature descriptors in literature use naive raw image intensities within the patches as descriptions. Some other approaches use filter responses over the image patch to reduce the dimensionality of the descriptor and store the filter responses, instead of the pixel values. Mikolajczk and Schmid [29] use steerable filters, which are Gaussian derivatives steered in the direction of prevailing gradient orientation in the image patch. Steering the derivatives makes the descriptor invariant to image rotation. One of the most popular approach which has motivated extensive research in the area is the use of SIFT descriptor.

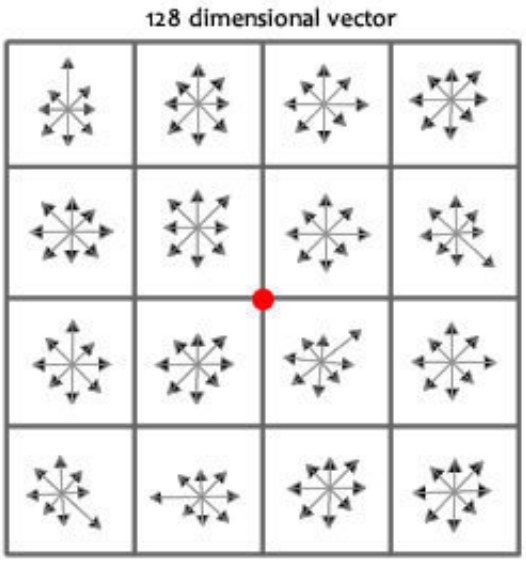



**Figure 2.1** Illustration of the SIFT descriptor. Image gradients within a patch (left) are accumulated into a coarse $4 \times 4$ spatial grid. A histogram of gradient orientations is formed in each gird cell. 8 orientation bins are used in each grid cell giving a descriptor a dimension of $128 = (4 \times 4 \times 8)$ [26].

**Figure 2.2** Two images showing a sampling of the SIFT features extracted in them.

**SIFT**: Lowe [26] proposed the gradient orientation histogram, which retains coarse spatial information. The SIFT (Scale Invariant Feature Transform) descriptor is illustrated in Figure 2.1. To achieve rotation invariance, all gradients within the patch are computed relative to a dominant gradient orientation, which is obtained as the highest peak in a histogram of all gradient orientations within the patch. Figure 2.2 shows a sampling of SIFT features extracted in two images from Golkonda fort in Hyderabad.

The SIFT descriptor has been experimentally shown [32] to outperform other descriptors like steerable filters [29], complex filters [41], and cross-correlations on raw pixel intensities, in the context of matching affine covariant regions. This is partially because of its high dimensionality (compared to, for instance, filter responses) and partially because it is tolerant (due to the coarse spatial grid) to small localization errors, which often occur. Thus the reason for this superior performance is that SIFT, unlike other descriptors, is designed to be invariant to a shift of a few pixels in the region position, and this localization error is one that often occurs. Combining the SIFT descriptor with affine covariant regions gives region descriptions vectors which are invariant to affine transformations of the image. Note, both the region detection and the description is computed on monochrome versions of the frame, color information is not used in this work.

## 2.2 Feature Matching and Bag-of-Visual-Words Model

Features extracted from various regions in the image can act as a proxy for the image to allow matching with other images. A rough quantification of image match between two images can be done simply by looking at the distance of the nearest neighbours of every feature in the first image in the second image. Thresholding the nearest neighbour distance would quantify the number of true matches


**Figure 2.3** Bag of Words in Text Retrieval: Significant words extracted from documents provide a quick impression about the documents. It can be easily inferred that the above two documents are not similar. Image credits : Li Fei-Fei [11]

in the two images. Additionally, normalization with respect to total number of features can also be done. A decision about whether the two images are matching can be made based on this.

Popular descriptors, such as SIFT, however are high dimensional(128 in this case), and hence, computing distances between features is an expensive process. Therefore, it is useful to quantize the feature space. This quantization first of all helps in reducing the amount of data that needs to be dealt with and also simplifies the matching: all features quantized to the same bin are deemed to be matching to each other. One popular implementation of this technique is the Bag-of-Visual-Words based matching, which is explained next.

### 2.2.1 Bag-of-Visual-Words based matching

The Bag-of-Visual-Words is a model inspired from Bag-of-Words model used in text retrieval techniques: If we compile a list of significant words, and compute the occurrence of these words in different documents, then the similarity of these documents can easily be inferred by counting the occurrences common significant words as shown in Figure 2.3. This concept was extended to Computer Vision by Sivic et al. [47] by quantizing the feature space of SIFT descriptors, and calling the quantization regions as "visual words", in correspondence to words in the text retrieval domain. The visual words are obtained by clustering, for example using k-means [57], a sample of SIFT features extracted from several images. Hence, visual words can be though of as the cluster centers obtained by this clustering. For obtaining the visual word representation of an image, all the SIFT features extracted in the image are

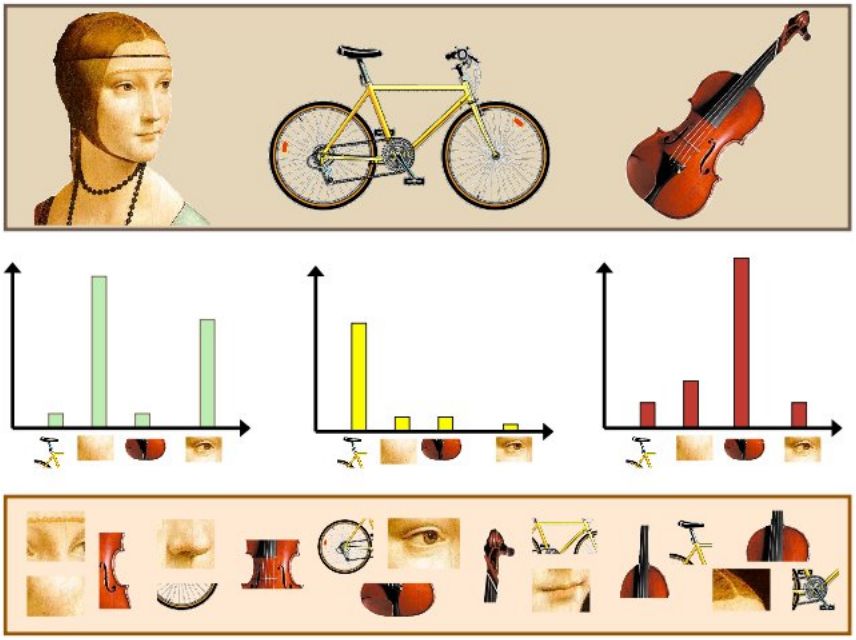


**Figure 2.4** Bag of Visual Words : Top row shows 3 sample images, the bottom row shows feature regions in the images, and the middle rows shows the histograms of the images with respect to 4 visual words. Image credits : Li Fei-Fei [11]

quantized to the nearest visual word, according to a distance metric commonly L2 norm, and a vote is given to the respective bin in the histogram. A histogram representation helps in the direct computation of the a matching score between two images by using a cosine similarity measure. Figure 2.4 shows a simplified representation of this concept.

### 2.2.2 Geometrical Verification

The Bag-of-Visual-Words model provides a compact summary of the image content, this however comes at a high cost of complete neglect of geometrical information in the image. As such the top results provided by the Bag-of-Visual-Words based retrieval may not always be correct. This makes it necessary to a geometry based verification to remove spurious matches. To do this, we try to find a consistent explanation for the visual word correspondences found in the two images. The correspondences of two images capturing a similar object can be explained by a fundamental matrix which maps a point in the first image to a epipolar line which corresponding to the ray produced by the back projection of the point in the first image into the other image. Figure 2.5 shows a point in one image and its corresponding ray which produces a epipolar line in the other image.

In general, as few as five point correspondences are sufficient to posit a fundamental matrix. The quality of a selection can be measured by the number of other correspondences, called inliers, which

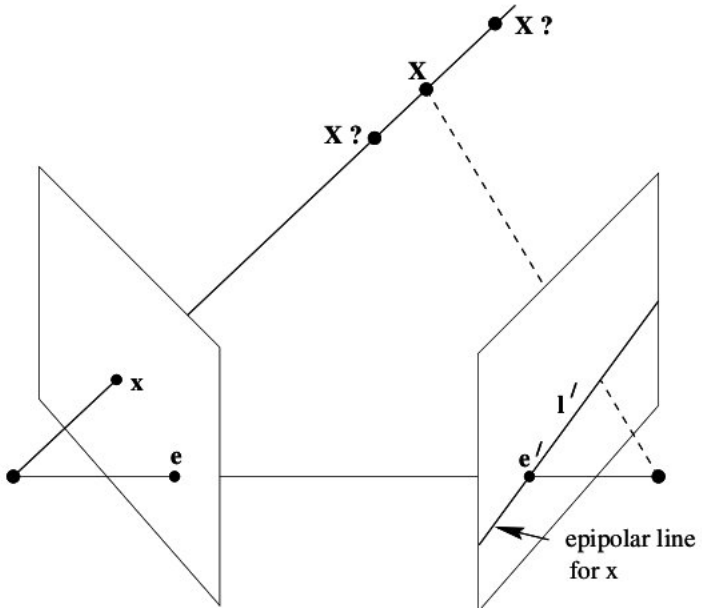



**Figure 2.5** Epipolar lines : The Back projection of the ray corresponding to point x produces the Epipolar line on the other image. Image reproduced from [17].

are consistent to the fundamental matrix. Even if one of the 7 correspondences is wrong, then the rest of the correspondences are very unlikely to correspond to the fundamental matrix obtained. an best explain the epipolar geometry of the matching features between the images. Therefore, we use RANSAC [12] algorithm which randomly selects and evaluate several candidate fundamental matrices. This procedure keeps track of the number of inliers obtained for the best candidate encountered. This maximum number of inliers obtained is used as the criteria for selecting or rejecting a match proposed by the Bag-of-Visual-Words model. Figure 2.6 shows a scenario where the top matches reported by the Bag-of-Visual-Words are verified to remove the spurious matches.

## 2.3 Mining image-matchs in Community Photo Collections

Large community photo collections are excellent resource to visually learn about a particular landmark or a destination. However, to build any useful application, it seems natural to first discover correspondences and common regions in images. A simple data structure to record this is a *Match Graph*, which encodes the pairwise relationships between the images. The images are represented as nodes, and then a match between two images is represented using an edge between the corresponding nodes. These match graphs can aid in various applications such as Browsing, Annotation Propagation, Object Mining etc. Next, we review the existing techniques for building match graphs.

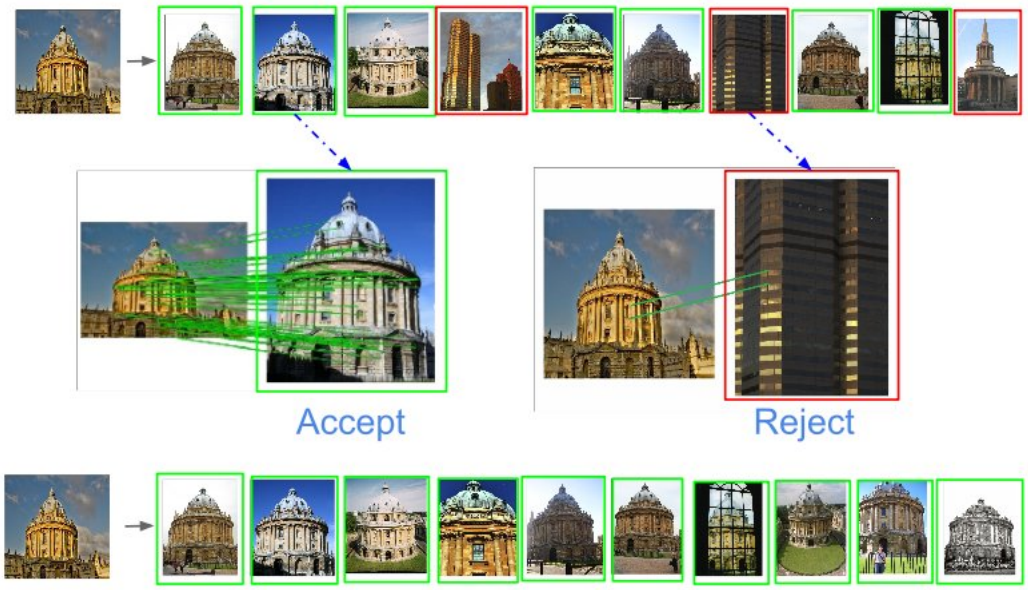



**Figure 2.6** Spatial Verification : Top row shows top results obtained for a query image by using a Bag-of-Visual-Words based approach. Correct matches are outlined in green and the spurious matches are outlined in red. Inliers obtained for a truly matching and a spuriously matching images are shown next. Spurious matches are eliminated to obtain more accurate top results.

### 2.3.1 Landmark Mining from match graphs

Philbin et al. [39] created a match graph on a large collection images( $>$ 1M) download by searching for "Rome" on Flickr and used it to mine Landmarks present in Rome. Their approach for computing matches was to *index* the Bag-of-Visual-Words based representation of every image in the collection, and then query each of the images via the index. The issue of indexing is addressed in detail in the next chapter. The querying was done in two states: First, 400 top ranked images were shortlisted and subsequently scored according to the number of geometrically verified inliers. An edge was created between the query image and its top 400 retrievals, and assigned a weight equal to the score computed above. Note that many of the edges in this graph my be spurious and may lead to coleasing of many different unrelated images. Therefore, connected components are discovered in this graph and they are subsequently over partitioned using a *spectral clustering* algorithm.

To re-establish connections in the over-partitioned clusters, a cluster merging approach is applied to re-join different clusters depicting the same object. This involves finding the images with the highest degree in each of the clusters. Next, a "query region" is then propagated via homography form these highest degree images towards other such highest degree images, using the shortest path computed from the original graph, to see if clusters can be merged.

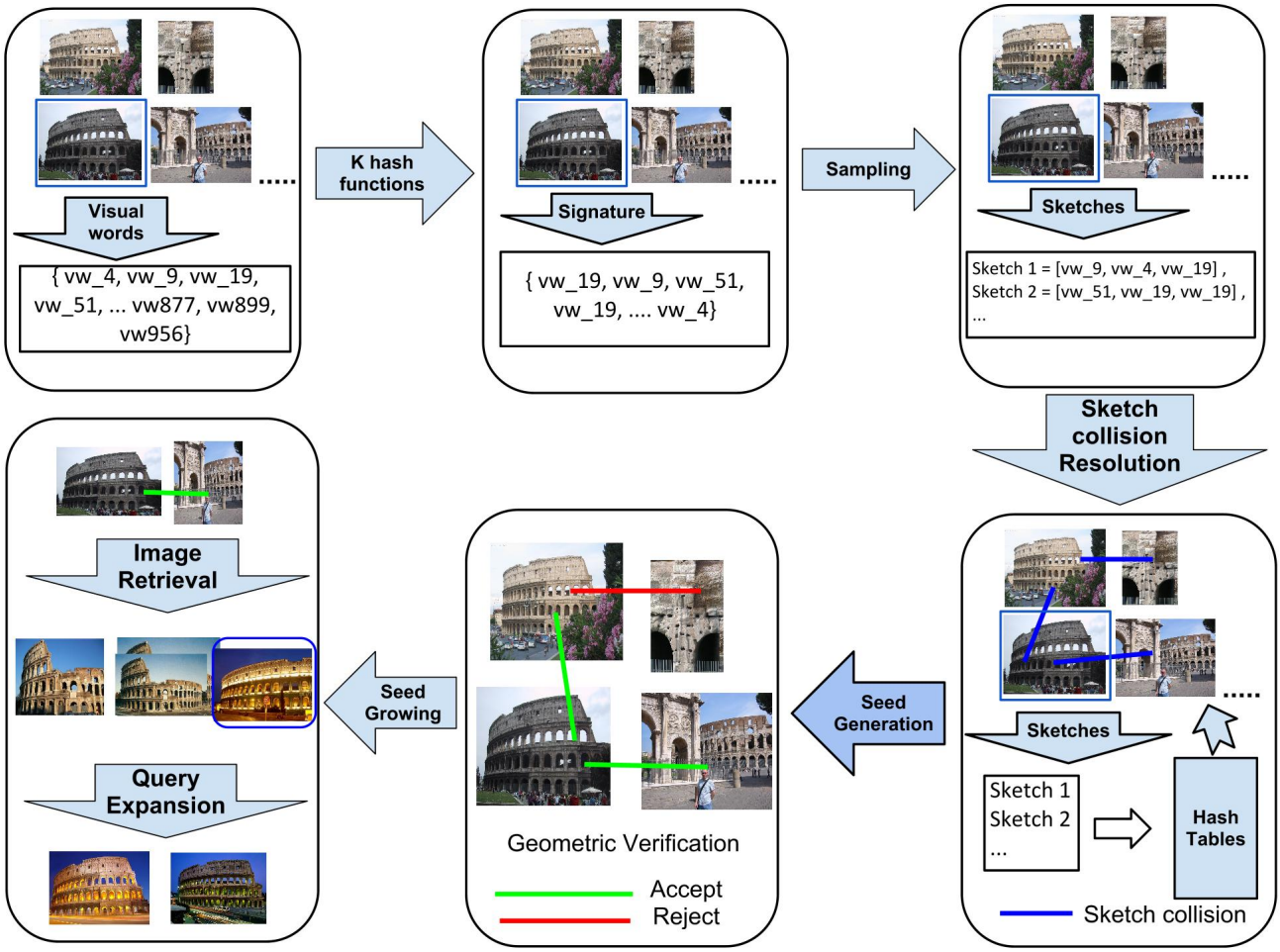



**Figure 2.7** Illustration of the MinHash Algorithm applied to a set of images by [8]

### 2.3.2 MinHash based Techniques

MinHash is an algorithm for quickly estimating the similarity of two sets, when the collection of possible elements is known and finite. It involves computing multiple hash functions over the sets to produce a signature of the set. These hash functions are designed to select the foremost element in different random permutations of the elements present in the set. This leads to a property that the probability of two sets having the same hash value is equal to their Jaccard similarity(J) [7]. Jaccard similarity between two sets, $A$ and $B$, is defined as the ratio of cardinality of the intersection to the cardinality of their union. So, in this setting, an image, which can be treated as a set of Visual Words, can be concisely represented by its corresponding list of hash values, known as signature. The ratio of hashes that match in the signatures of two images acts an unbiased estimator of their similarity.

Chum et al. introduced [8], minHash based techniques for mining clusters of matching images by introducing a set representation of the visual words present. The set representation is created by binarizing the corresponding the histogram of visual words, that is, only the presence of visual words is noted, not the count. Up to 512 minHash function are evaluated on the images in the database to produce signatures. Next, a tuple of length 3, called a sketch, is computed by sampling 3 hash values in the signatures, and inserted into a hash table. All the sketch collision produced in the hash table are considered potential matches and their corresponding images are geometrically verified. A geometrically verified match thus found is called a *seed*, as they are entry points for discovering a cluster of matching images. For growing the seed, *query expansion* is used, which involves repeatedly querying the newly retrieved matches. The querying is done by indexing the Bag-of-Visual-Words representation of the images in the database, identical to Section 2.3.1.

### 2.3.3 Image Webs

Heath et al. [18] present a match graph building approach having two distinct steps: (a) Discovering connected components, and (b) Boosting connectivity of the connected components. The first step involves determining, for every image in the database, the top $K$ matching images according to the Bag-of-Visual-Words model in a manner similar to Section 2.3.1. These matches become candidates for doing an expensive cosegmentation procedure, which segments a common object in a pair of images, and are sorted in the order of their match scores. The cosegmentation procedure, similar to geometric verification, also determines whether two images match or not. Since, the goal of the first step is only to determine the outline of the connected components, only the candidates which lie in two different components undergo cosegmentation.

The next step is to individually boost the connectivity of each of the connected components discovered. For this, each of the connected components are treated like a new database, and the top $K$ matches for each images are determined in their respective connected components, just like in the first step. The sorting of the candidate matches for doing cosegmentation is however different. For this, a Laplacian matrix for the images in a connected component is built according to the match scores, and the candidate match which has the highest potential for improving the *algebraic connectivity* of the matrix is selected for cosegmentation.

## 2.4 Photo Tourism

Photo tourism, created by Snavely et al. [53], is a graphical system for browsing large image collections in 3D which combines image-based rendering and photo navigation techniques. This approach can handle large input image collections of images about which no prior information is given. This approach automatically does a sparse 3D reconstruction of the scene from which the photos were taken as a point cloud. It also provides the configuration of the cameras which have taken these photos. The

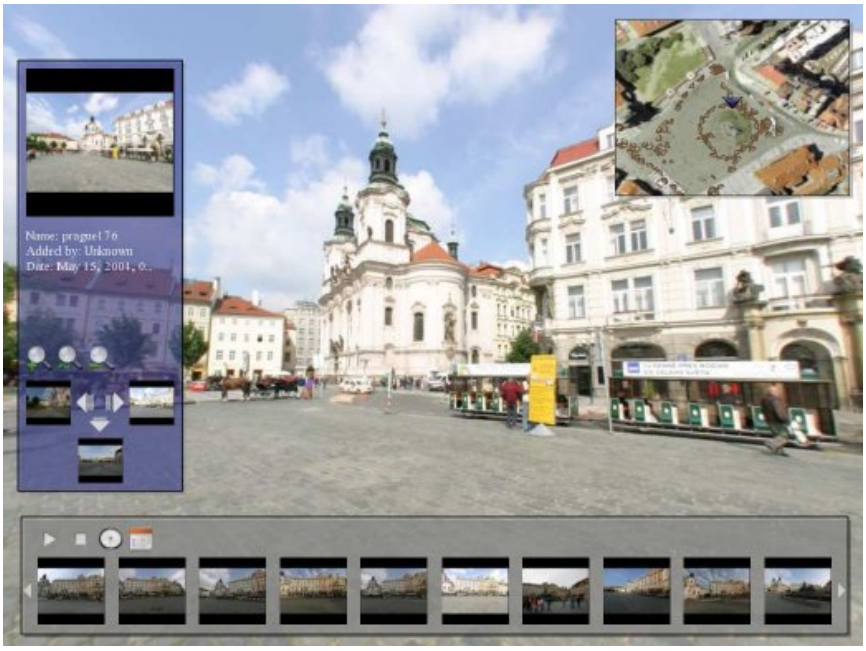



**Figure 2.8** Illustration of the relation based browsing features for scene exploration in Photo Tourism.

underlying principle for Photo Tourism is to scale Structure from motion to large sets of image about which no prior information is known. Thier struture from motion pipeline called bundler works by carefully selecting a seed pair of image to initialize the reconstruction. Followed by incremental addition of matching images which in small batches. After each such iteration, a refinement is done to the estimated parameters inorder to minimized the error using a expensive non linear optimization technique called bundle adjustment. The final output is the a sparse 3D representation of the scene, as well as an estimate of the camera parameters of the input images.

The photo explorer interface allows the interactive exploration of the scene in the computed 3D point cloud space using free flight navigation, also, the images are also registered on a overhead map. This space also shows the input images used for creating the reconstruction by placing them in accordance with the camera parameters computed earlier. A users can do a variety to interactions to simultaneously explore the scene and the image collection such as finding a zoomed out version of the image, finding a image to the left of the image, etc. to do relation-based browsing as shown in Figure 2.8. A user can also do object based browsing by selecting a region of an image and the system then takes the user to a list of images which show the object in more detail.

This seminal work on community photo collections was paved way for doing Multi View Stereo [16], summarization [43], finding paths [51] and skeletonization [55] of image collections. At the same time, effort was also put in improving bundle adjustment for large image collections [1], [10]. The point clouds and 3D models extracted from Community photo collection were also used for doing location

recognition of images. In [2], matching and reconstruction algorithms were parallelized and distributed at each stage in the pipeline to reconstruct a the city of Rome with 150 K images in less than a day on a cluster with 500 compute cores. In [23], Li et al. proposed a novel clustering based approach to using the low dimensional global descriptors to extract iconic images which share features with all the other images in the cluster. The relationship between the iconic views is captured by an iconic scene graph which not only acts as a summary, but can also be used to direct the Sfm the efficiently produce the 3D reconstruction. In [13] a highly parallel implementations of image clustering, stereo, stereo fusion and structure from motion were deployed on modern graphics processors and multi-core architectures to reconstruct Rome with 3 million images within the span of a day on a single PC.

*Chapter 3*

# Fast Indexing and Retrieval in Large Image Collections

The key to matching images, lies in matching features extracted in the images. The challenges for doing this originate mainly from (a) the large size of image collections, and (b) high dimensionality of features leading to a high cost of distance computation and large amount of main memory requirement to store the feature representation. The Bag-of-Visual-Words model solves the second problem by quantizing a feature to a visual word which can be denoted by a simple datatype such as an unsigned integer. This representation also bypasses the need for doing any sort of distance computation as only the features quantized to identical visual words are considered matching.

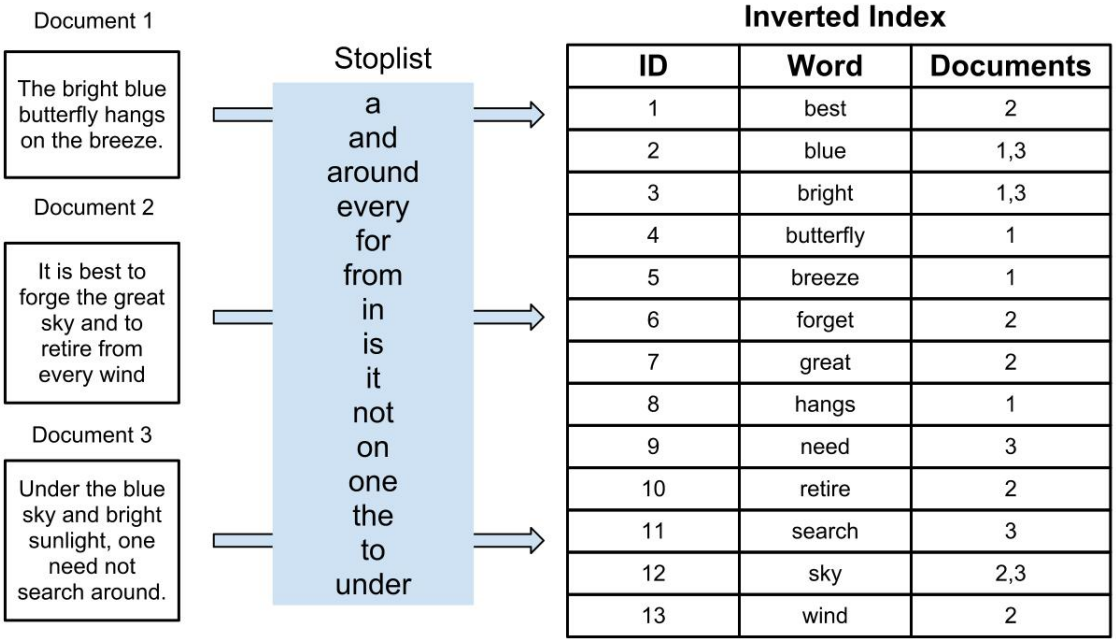



**Figure 3.1** Inverted index on a database of text documents.

For querying a large collection of database images to find images similar to a query image, it would be a very time taking job to be comparing the visual word histogram of the query to the histograms of all the images in the database. Given that histograms are usually sparse, what would really be useful is to quickly find out which images have at least one visual word in common with the query image. So an index, called inverted index, mapping visual words to the posting list of images in which it occurs is built. This index is kept in main memory to provide fast retrieval given a visual word query.

For scoring, initially an empty score array is initialized. Then for every visual word in the query, score is incremented for all the images appearing in the posting list in the inverted index. Concepts such as term frequency and inverse document frequency, dubbed *tf-idf*, may be used to provide normalization with respect to the number of features in every image, and to downplay the role of commonly occurring visual words. Additionally, a list of *stopwords* can be created to prevent heavily occurring words contributing to the matching of 2 images. Figure 3.1 shows an a inverted index built over the words occurring in a database of 3 documents. Note that heavily occurring words such as "a", "an", "the" etc. are included in the list of *stopwords* and are not indexed. Figure 3.2 shows a similar inverted index built over the visual words occurring in a database of 3 images.


**Figure 3.2** Inverted Index over the visual words occuring in a database of images. Image credits : Kristen Grauman.

In spite of providing a direct access to the list of all the images which contain a visual word in the query image, Bag-of-Visual-Words based retrieval using an inverted index technique has some limita-

tions which affect its performance in handing large image databases. First, this technique inherits the inability to capture any geometrical information present in the image. Therefore, geometrical verification needs to be applied separately. Since, this operation is expensive and requires access to the relevant features in candidate images, this process can only be applied only to a small number of candidates images. This may lead to missing out of several matching images which ended up further behind due to low matching score. Secondly, prior to the verification of the shortlist, the number of images which need to be considered in the posting lists, and hence, in the score array, is proportional to the size of the database. Therefore, though it can be argued that the time taken for querying a image is constant [39] in practice, but the actual time complexity, as also noted in [8], is linear in the size of the database.

## 3.1 Towards faster Indexing and Retrieval

The scalability issue with inverted index arises because of its constant length, which is determined by the number of visual words. It can be seen that, if the length of the index could be chosen in proportion to the size of the database, then the average size of the posting lists will be constant, leading to constant time retrievals. However, one cannot arbitrarily increase the number of visual words without severely affecting the matching performance. This effect was observed in [38], where the mean average precision of querying the Oxford Building Dataset dropped while moving from a vocabulary of size 1M to 1.25M.

To tackle this problem, we use *High Order features* created by combining nearby features(visual words). As a result, geometrical information will get embedded into the index. A similar notion of spatial consistency was also used in [48] where the number of correspondences of neighbouring features of two matching features was used to measure the confidence in the matching features. The use of *High Order features*, however, raises issues regarding building an inverted index of the extremely large domain of the High Order features. The next section shall deal with this issue in detail.

## 3.2 Indexing over High Order Features

We define, High Ordered features as the features which appear in the close vicinity of each other. For computing them, we determine the $n$ nearest neighbours(*secondary feature*) of every feature(*primary feature*) which are closer than a distance of $m$ pixels. It is represented as a tuple of the visual word of the *primary feature* followed by the visual word of the *secondary feature*. Given the cardinality of domain of visual words as $C$, the cardinality of the domain of *High Order features* would be $C \times C$. For example, given a vocabulary of size $10^6$, then corresponding domain of High order features would be $10^{12}$. Given this extremely large domain, an inverted index build over the domain of high order features would have extremely sparse posting lists, even for very large databases. Unfortunately, the extreme size of the domain does not allow the allocation of a corresponding inverted index. As we shall see next, a Bloom filter based index can provide us with the benefits of using an extremely large domain, while not requiring an inordinate amount of memory for allocation.

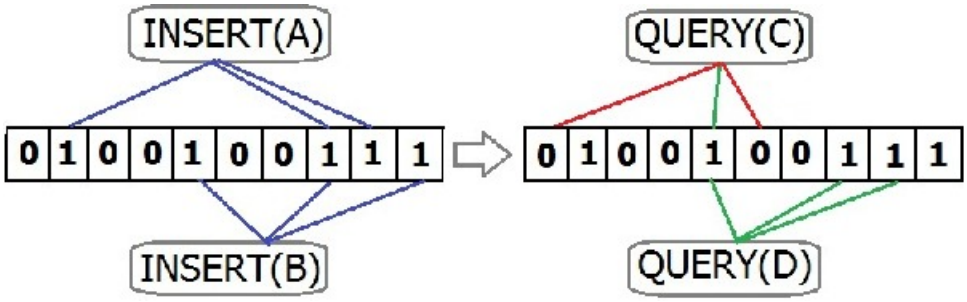



**Figure 3.3** Bloom Filter: [left] The process of insertion and querying in a Bloom filter of size 10 bits with 3 hash functions. [right] The first query is is not present in the Bloom filter. The second query is a false positive.

### 3.2.1 Bloom filter

Bloom Filter [6] is a space efficient data structure for doing set membership queries. It allows constant time insertions and set membership queries. The downside of this scheme is that it occasionally identifies a non member query element as present. These are called *false positives*. Hence, the output for each query can be interpreted as:

- $1 =$"present in set, but can be wrong"
- $0 =$ "certainly not in set"

Also note that, for a set $X = \{a, b, c, d\}$, the elements of which are inserted into a bloom filter, say $B$, if a query is made about the element $a$, then $B$ would defiantly return that it is present in the set. One definite advantage of Bloom Filters over Hash Tables is that it allows the addition of new elements to a certain limit without any additional need for memory. It is interesting to note that a Bloom filter does not store the actual data, it only works on the hash values produced by the data. The price to pay, however, for overfilling a Bloom Filter is an increased false positive rate while querying. Bloom filters and its variants are recently being used in many tasks such as data redundancy removal [5], sensing of networks [63] etc.

#### 3.2.1.1 Implementation Details

Let us look at the implementation details of a Bloom filter. A Bloom filter, $B$, is composed of a bit array, $A$, and a fixed set of associated hash functions, $H$. For inserting an element, $e$, into the set $S$ being represented by $B$, all the $h \in H$ are evaluated for $e$, and the bit positions corresponding to the resulting hash values are set to 1. For doing a membership query, $h \in H$ are evaluated on the query element, and all the bit positions corresponding to the resulting hash values are checked for their set value. If any one of these bit positions is not set in the bit array, then the element is definitely not in the set, otherwise the element is deemed present. A false positive occurs when *all* the bit positions corresponding to the hash values of a query element have already been set by other elements inserted before.

| m/n | k=1 | k=2 | k=3 | k=4 | k=5 | k=6 |
|---|---|---|---|---|---|---|
| 2 | 0.393 | 0.400 | | | | |
| 3 | 0.283 | 0.237 | 0.253 | | | |
| 4 | 0.221 | 0.155 | 0.147 | 0.160 | | |
| 5 | 0.181 | 0.109 | 0.092 | 0.092 | 0.101 | |
| 6 | 0.154 | 0.0804 | 0.0609 | 0.0561 | 0.0578 | 0.0638 |

**Table 3.1** False positive rates(FPR) for various combinations of parameters $m$(size of the Bloom filter)$/n$(number of elements to be indexed) and $k$(number of hash functions).

Given the number of hash functions($k$), the size of the bit vector($m$) and the number of elements($n$), it is possible to determine the false positive rate by: $(1 - e^{-kn/m})^k$. Figure 3.3 shows a simplified representation of Bloom filter operations. Table 3.1 shows false positive rates for low values of $m/n$ and $k$. As intuition would suggest, it can be seen that the false positive rate becomes less when a larger bloom filter is used, that is, $m/n$ is increased. However, increasing the number of hash functions only improves the false positive rate only upto a limit. This limit can be computed as $ln2 \times m/n$. Crossing this limit should always be avoided as this implies more hash value computation for bad false positive rate.

#### 3.2.1.2 Toy Example for Bloom Filter

Let us consider a bloom filter, $B$ of size $M$, in which multiples of 7 less than 1000, a total of 142 elements($N$), are to be inserted. We use 2 simple linear hash functions to do these operation as shown in the C++ code below.

```
int hashfn1(int x){
        int a=4754,b=733;
        return (a*(x/10)+b*(x%10))%bfsize;
}
int hashfn2(int x){
        int a=3455,b=587;
        return (a*(x/10)+b*(x%10))%bfsize;
}
```

Table 3.2 lists the performance of this Bloom filter for various values of $M$ while querying the first thousand Natural numbers.

| $M$ | $M/N$ | Queries | Set elemets | #False Positives | Sample False Positives |
|---|---|---|---|---|---|
| 100 | 0.71 | 1,2,..,1000 | 7,14,21,..,994 | 858 | 2 3 4 5 6 8 9 10 11 12 13.. |
| 200 | 1.42 | 1,2,..,1000 | 7,14,21,..,994 | 696 | 2 3 4 5 6 8 9 10 11 12 13.. |
| 400 | 2.85 | 1,2,..,1000 | 7,14,21,..,994 | 233 | 3 5 8 12 15 18 19 22 24 26.. |
| 800 | 5.71 | 1,2,..,1000 | 7,14,21,..,994 | 50 | 3 12 15 24 68 80 101 122 135.. |
| 1600 | 11.43 | 1,2,..,1000 | 7,14,21,..,994 | 5 | 15 24 279 338 681 |

**Table 3.2** False positives obtained for various sizes of Bloom filters using 2 Hash functions.

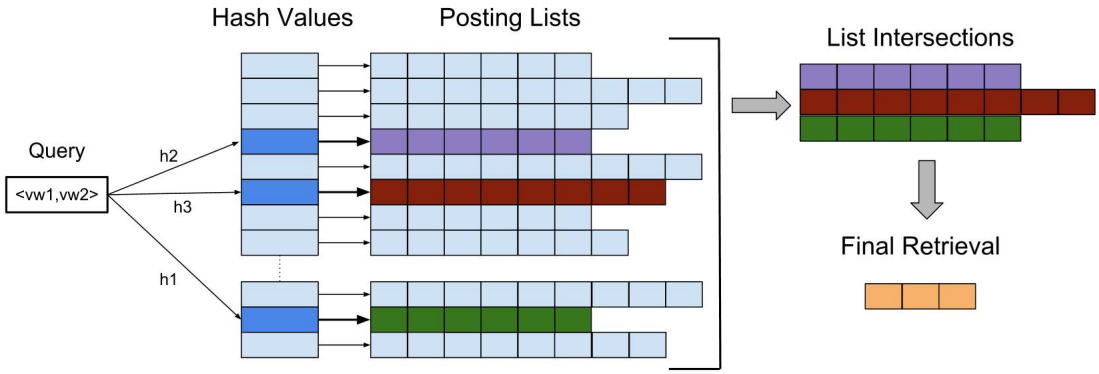



**Figure 3.4** Inverted index over Bloom filters : Common images present in the posting lists corresponding to the hash values of the query high order features are given a vote.

### 3.2.2 Inverted Index over Bloom Filters

A simple indexing scheme can be built for $N$ images by allocating N Bloom filters, $A_1 \ldots A_N$, of identical size and using identical hash functions, and inserting the respective high order features of each of the $N$ images. A query high order feature, $q$ can now be resolved by evaluating all $h \in H$, and checking the corresponding bit positions in all the Bloom filters. It is easy to see that this process can be sped up by storing the bit arrays of the Bloom filters as an inverted index over the bit positions.

Retrieval can now be done easily by taking intersections of the posting lists of bit positions corresponding to evaluations of all $h \in H$ on $q$. This operation can be executed very fast as it is possible to store the posting lists in a sorted manner without any additional overheads. Next, score is incremented for the images containing the query high order feature. Figure 3.4 illustrates the process of querying high order features in the query image using an inverted index over the Bloom filters of the database images.

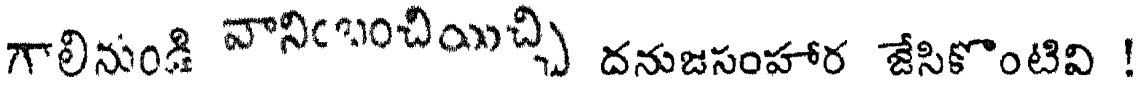


**Figure 3.5** Some of the word-images taken from the Telugu dataset.

## 3.3 Experiments with Word-Images

A word-image is an image of a printed word. These can be obtained by scanning a printed document followed by segmentation of word regions. In order to demonstrate the feasibility of our technique, we do image retrieval on a database of Telugu word-images. The Telugu word-image dataset, has 4304 images of words written in Telugu script extracted by scanning the individual pages of a book. Figure 3.5 shows a few of these images.

We prepare the ground truth by clustering all the instances of every word in the database. Given an image of a word as query, the retrieval system should ideally return rest of the instances of this word in the database. We have used 15 such queries to test our approach. We have used Harris-affine detector to detect interesting regions in word images. These interesting regions were described by the SIFT [26] descriptor. We created a vocabulary of 24000 visual words by hierarchically clustering [37] the SIFT descriptors extracted in the 4304 images. We extracted high order features, and computed two hash functions per high order features corresponding to the bloom filters of each of the images. Next, we indexed the bloom filters by building an inverted index of size $2^{29}$ over the bloom filters as described in section 3.2. Using the approach were able to obtain a mean average precision of 0.6 on the dataset for the 15 queries. The average time taken for each query was 0.6 micro seconds. Figure 3.6 shows retrieval results for some of the query images. Note that some of the matches have also occured due to matching characters.

## 3.4 Discussion

In this chapter we see that building an inverted index over the set of visual words speeds up the querying. This speed up is controlled by the number of visual words used to quantize the features extracted in the database images. It is however not practical to obtain any desired speedup by simply increasing the number of visual words, as after a limit it severely affect the matching of features. We provide a solution to this problem by producing a much richer quantized features space by combing nearby visual words as High order features. This huge features space is projected to desired size using hash functions, and an inverted index is build over this space to provide the desired speed up. Our experiments with word images validate the usefulness of our approach. We show the application of this technique to Community Photo Collections in the next chapter.

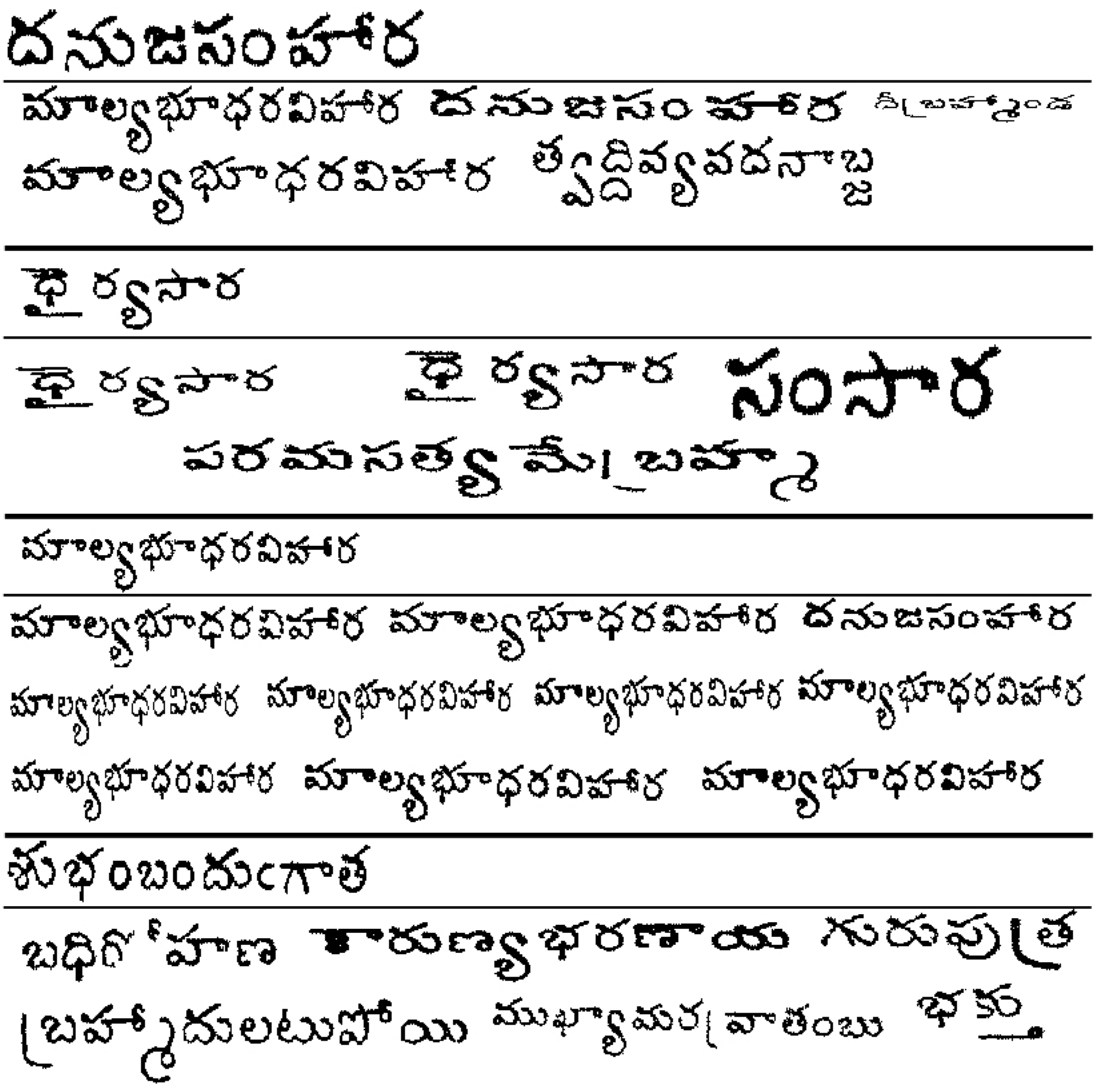


**Figure 3.6** Query results of some the word-images from the Telugu dataset. The query word is shown on the top and the reterieved results are shown under it, below the light horizontal separator.

*Chapter 4*

# Towards Fast Match Graph Construction on Image Collections

The easy accessibility of large collections of images has opened opportunities for mining them. These datasets are excellent resources for location recognition[24], browsing[18], summarization[44], reconstruction[53] or creating walkthroughs[56] of various popular destinations. Given the current techniques for harvesting these large collections, they tend to be highly unordered and have a lot of irrelevant images. Hence, the automatic organization of these datasets is very much required. We present a method to organize these datasets as image match graphs. A match graph is a simple data structure in which images are depicted as nodes, and a match between images is represented by an edge. This representation allows the discovery of interesting intermediate images to connect distant images, as shown in Figure 4.1, and small clusters of matching images.

The match graph construction problem is to produce a graph denoting matches between any pair of images in the dataset. This problem is related to the image retrieval problem, where the user supplies a query image and the system returns a ranked list of similar images. Many of the recent techniques [38] for image retrieval utilize the Bag-of-Words(BoW) framework to implement a filtering stage whereby a large number of images are rejected. In this framework, visual words are cluster centers extracted by clustering a large collection of descriptors extracted from various images. These visual words are used to quantize feature descriptors, such as SIFT [26], SURF [4] etc. To speed up the matching process, an inverted index is built which maps the visual words to a posting list of images which contain them. For every visual word in the query image, a vote is given to all the images which contain that visual word. A shortlist is obtained by taking into account the top scoring images. A geometric verification is performed on the images in the shortlist for reranking and rejecting non matching images.

Many of the recent techniques for image retrieval [65, 61, 9] try to bring geometry into the filtering stage to obtain more precise posting lists. Zhang et al. [65] use the geometry preserving visual words which captures both cooccurrences, and local and long-range spatial layouts of the visual words to outperform even the BOW model using RANSAC for geometry verification. Similarly, [61] incorporate the neighborhood statistics of features into the vocabulary tree and in the spatial domain to improve the discriminative power of the features.

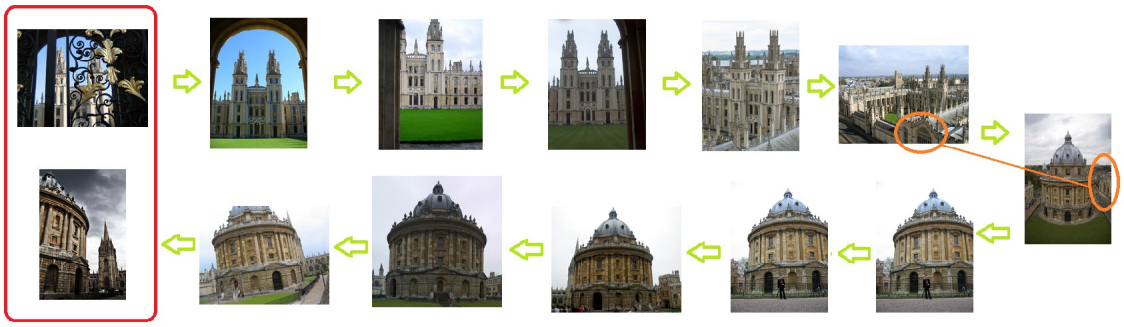


**Figure 4.1** A path discovered from an image of All Souls Building to an image of Radcliffe Camera in Oxford in our match graph.

Efficient solutions for match graph construction also use techniques use for efficient image retrieval, such as feature quantization, indexing etc., to limit the amount of data that needs to be dealt with. Image retrieval also provides an immediate solution to the match graph construction problem: 'query each image in turn' and create a link from every image to each of its verified retrieved images [39]. Query expansion is used on these verified images to discover more overlapping images. Similarly, *Image webs* [18] use Image retrieval techniques to identify the skeleton of clusters, and complete the clusters by verifying potential links within a connected component with a focus on maximizing the *algebraic connectivity* of the cluster.

Chum et al. introduced Minhash based techniques [8, 9] which employ random sampling of the visual words using MinHash functions to obtain image signatures, similar to image histograms in the BoW framework. The signatures are sampled to obtain sketches, which become more discriminative than an individual visual word. Hence, all the sketch collisions, which are detected using a hash table, are verified. This makes the chance of discovery of a matching pair of images independent of the size of the database. The verified matches, called seeds, are grown using query expansion to obtain full clusters. One disadvantage of this technique is that the chance of discovery of small clusters is not very high.

Notwithstanding, the success of the above techniques, it can be seen that the ideal solution for match graph construction is matching every image in the database to every other, that is, exhaustive pairwise matching. We explore the feasibility of doing this for building match graphs for large datasets. For this, we build upon the advantages of the above techniques: First, we employ an inverted index based retrieval scheme which provides direct access to the list of relevant images for a given query. Second, similar to the sketches used by Minhash based techniques, we use high ordered features, extracted from pairs of nearby features, to do feature matching in a more discriminative space. The indexing of geometry allows us to do match verification directly from index retrievals. We are thus able to implement 'query each image in turn' for exhaustive pairwise matching in linear time complexity.

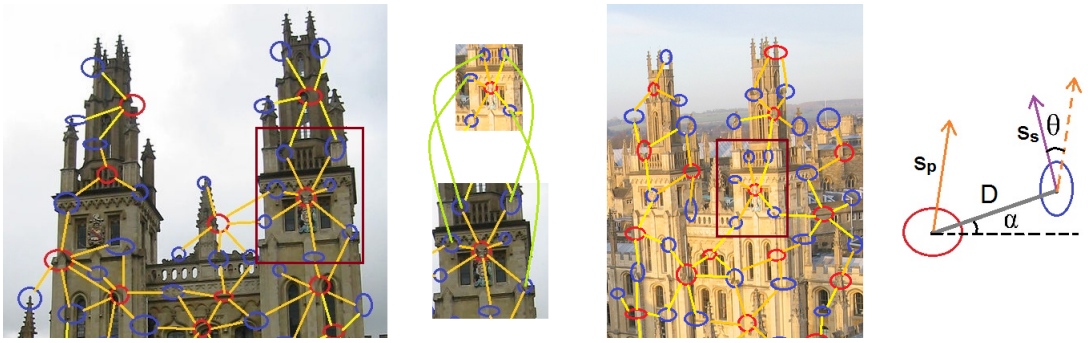



**Figure 4.2** A sampling of High Order Features(yellow) extracted in a pair of images. Geometric parameters( $s_p/s_s$ , $D/s_p$ , $\alpha$ and $\theta$) are computed for a primary feature(red) with respect to all its secondary features(blue). $s_p$ and $s_s$ denote the scales of the primary and secondary features respectively. Vectors point towards the dominant orientation of the features. Four matching high order features are shown(green) which correspond to the primary features highlighted in the images.

## 4.1 High Order Features for Exhaustive Pairwise Matching

The direct implementation of 'query each image in turn' for building match graphs is not applicable due to its quadratic matching cost. In [39] and [18], the number of images needed to be taken into consideration for a query visual word is proportional to the size of the database. The number of such queries needed to be issued is proportional to the size of the database, making the overall timing complexity of the whole process quadratic in the number of images. Also, the need for keeping a shortlist of candidates from the filtering stage has the potential to miss out some of the matching images, thereby missing the quality of exhaustive pairwise matching. This issue becomes serious in the presence of a large number of distractor features, like those coming from trees, water etc., which dilute the contribution of visual words coming from the object in the image. This leads to relevant images not being able to make into the shortlist.

Min Hashing based techniques[8] are able to bring down the computational cost by computing *sketches* which lead to lesser number of random matches than visual words. However, at a time, only one sketch per image is taken for matching as compared to a histogram level matching in [39, 18]. This affects the chance of discovery of matching images having only a few visual words in common. The chance of discovering a match also depends upon the size of the sketch. Choosing a low sketch size would find many matching images, but would also lead to many irrelevant sketch collisions in large datasets. In [8], using a sketch size of 3 for Oxford 100K Oxford Landmark database, 38.4 sketch collision were generated per image which lead to only 441 verified seeds in total. Therefore, a high sketch size is used in practice, but this leads to missing out seeds in smaller clusters and in clusters having low average image similarity. Moreover, since query expansion is used for completing the clusters, the success of these techniques is indirectly affected by the size of the database. The effect of these phenomenon is observed in the cluster representing the landmark "Magdalen Tower" in the Oxford

Buildings Dataset [38], where only 3 of the 54 valid images were discovered, and no other image could be discovered through query expansion.

We identify scalable exhaustive pairwise matching as a feasible paradigm to overcome aforementioned issues. It can be seen that the inverted indexing technique could be made scalable if the size of the inverted index could grow with the size of the database, making the average size of the posting lists constant to allow querying in constant time. This is, however, not possible given the fixed domain of visual words. Therefore, we extract high order features by combining a feature with its nearby features and encoding their respective geometric configuration. This provides an extensive domain which is much more discriminative than visual words, and can be easily reprojected to a required size, using hash functions, based on the size of the database to obtain constant average size of the list. Zhang et al. have used a similar notion of high order spatial features in [64] to find all the cooccurring feature occurrences under translation in a pair of images.

To address the problem of missing potential matches outside the shortlist, we design a match verification criteria which can be implemented on-the-fly from the index retrievals. This works well in practice as our high order features capture geometric information. Moreover, we match all high order features in a query image with all other high order features in the database, unlike [8], where only one sketch is pooled at a time per image for matching. This greatly enhances the chance of discovering small clusters.

### 4.1.1 Extracting High Order Features

We extract Hessian Affine regions and compute SIFT [26] descriptors of these regions for all the images in the database. These SIFT descriptors are quantized to a visual word vocabulary, using a kdtree built over the vocabulary. We choose nearby features for creating high order features, as their perspective projection into a matching image can be well modeled using a much simpler affine geometry. Next, we bin every image into bins of size 100 pixels. For each bin, we select up to 30 features, called *primary* features, by shortlisting features with the highest value of the scale parameter. Each *primary* feature is paired with upto 20 of their nearest neighbours, within a radius of 80 pixels, to create high order features. Since our high order features are confined to local regions, we expect their correspondences to follow affine geometry. Next, we compute the geometric parameters corresponding to the four grammar rules enlisted in [62]. Figure 4.2 gives a detailed description of these parameters. These grammar rules define invariants with respect to affine transformations. A high order feature is represented as a tuple enlisting visual words of the primary and the secondary feature, and the quantized values of the geometric parameters.

## 4.2 Indexing High Order Features using Bloom Filters

The domain of high order features is obtained by the cross product of the domain of visual words with itself and the domain of geometric parameters. Given a 1M vocabulary, the size of the domain of

our high order features is $10^{12}$ even while neglecting the geometric parameters. This makes it possible to reproject our domain to a custom size, in accordance with the size of the database, to obtain sparse posting lists in the inverted index. To do this, we design an indexing scheme inspired from Bloom filters which minimizes the memory usage and provide constant time retrievals.

### 4.2.1 Indexing using an Inverted Index over Bloom Filters

A simple indexing scheme can be built for $N$ images, by allocating equally sized Bloom filters, $A_1, \ldots, A_N$, with identical hash functions, and inserting the respective high order features of each of the $N$ images. A query high order feature, $q$ can now be resolved by evaluating all $h \in H$, and checking the corresponding bit positions in all the Bloom filters. It is easy to see that this process can be speeded up by storing the bit arrays of the Bloom filters as an inverted index over the bit positions. Retrieval can now be done easily by taking intersections of the posting lists of bit positions corresponding to evaluations of all $h \in H$ on $q$.

In the above framework, it is interesting to note that, keeping $k = 1$, that is, using only a single hash function, would eliminate any need of computing list intersections, making the retrieval process similar to that of BoW framework. However, unlike standard BoW, there is no restriction on the size of the inverted index, which is determined by the size of the bit array used. We can thus control the size of the inverted index in accordance to the size of the database to ensure a constant average length of the posting lists. This ensures constant time query retrievals. Use of only a single hash function was also used by Mitzenmacher in [34] to make Bloom filters compressible.

It is important to take care to always choose a big enough size of the inverted index so that the false positive rate while querying is low. Figure 4.2.1 shows the variation of false positive rate with $m/n$ for $k = 1$. For a given maximum number, $c$, of high order features for any image in the database, it is easy to see that a false positive rate better than $10^{-3}$ can be easily achieved by allocating an inverted index of size $1000c$. Assuming 20k high order features in a query and database images, this implies generation of $20 \times 10^3 \times 10^{-3}$ = 20 false matches on an average with every image.

### 4.2.2 Spatial Verification

We consider two primary features in different images a true correspondence only if they have at least $v$ high order features originating from them in common. This criteria can be evaluated directly from inverted index retrievals, by querying all the high order features from a primary feature in succession and selecting images which claim to have $v$ of these features. We consider an image level match verification to be passed if $w$ such truly corresponding primary features are found. This works well in practice as the high order features come from a very discriminative domain, resulting in very few mismatching of primary features across images. We keep the maximum number of secondary features per primary feature high at 20, to provide sufficient redundancy for finding common high order features with other primary features.

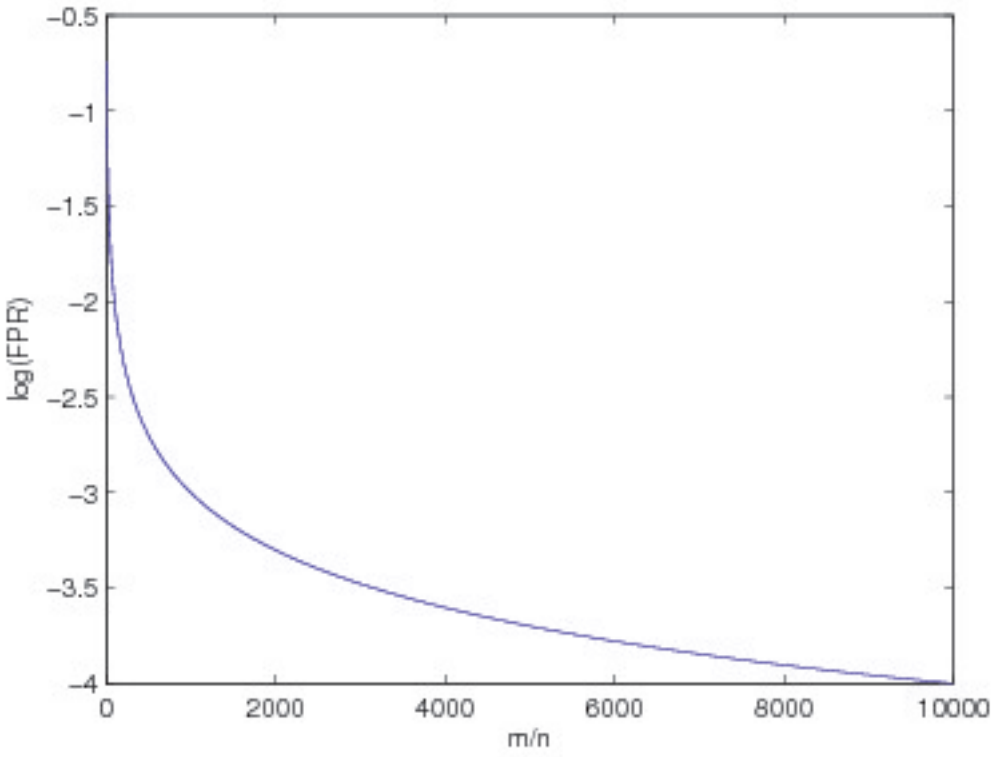



**Figure 4.3** Plot showing the variation of $log(FPR)$ with $m/n$ for one hash function($k = 1$).

Given the susceptibility of our retrieval scheme to errors, it is necessary for our spatial verification criteria to be robust to occasional mismatches. It is not difficult to see that our spatial verification scheme is robust to the introduction of a few spurious matches, as a single high order feature mismatch at random is not very likely make the corresponding query primary feature truly correspond to another in a non matching image. Similarly, the event of multiple spurious high order mismatches corresponding to a single primary feature is also unlikely.

### 4.2.3 Match Graph Construction

We start by choosing an appropriate size, $m$, of the inverted index as discussed earlier. For Match Graph construction, we use a three-pass strategy, where we start by declaring a counter array of the size $m$ to keep a count of high order features getting a certain hash value. In the first pass, we compute high order features and their corresponding hash value for all the images, while also updating the counter array. Now, precise memory allocations can be made for the posting lists of the inverted index based on the counter array. In the second pass, we index all the images by inserting their hash values computed

**Figure 4.4** Two of the objects retrieved by our method for creating match graphs on the UKBench Dataset.

earlier into the inverted index. In the final pass, we query each image in turn, and note down all the correspondences in an adjacency list.

## 4.3 Results

We use a standard image retrieval benchmark dataset, the University of Kentucky dataset(UKBench), introduced by Nister et al. [37], to measure our detection rate in small clusters. This dataset has 4 images each of 2550 objects making a total of 10200 medium resolution images. We used a vocabulary of 100K visual words for this experiment. On an average, 1048 features were extracted per image from this databset. An average of 413 primary features were selected for every image, which resulted in an average of 7436 high order features per image. This implies every primary feature combines with 18 secondary features on an average. A total of around 76m high order features were indexed using a simple FNV hash function. We defined the size of the inverted index to be $2^{25}$, while the highest number of high order features in a image was 23598. This corresponds to a maximum expected false positive rate of querying an image at $7\times10^{-4}$. This implies a maximum of 16.6 mismatches are expected to be generated between any pair of images.

The timing breakdown of the whole execution is as follows: Extraction of the high order features and computation of their hash value took an average of 0.1 seconds per image. Building the inverted index over Bloom filters took 39 seconds. Querying the database took around 0.073 seconds per image. The total time required for the whole process was 23.6 minutes. The number of bytes required for indexing is equal to the sum of size of the inverted index and the number of high order features, that is, $8 \times 32 + 4 \times 76 = 560$Mb.

For the purpose of measuring our efficiency in detecting small clusters, we choose our match verification criteria as finding one primary feature with 3 high order features identical. We were able to find 1868 object clusters, corresponding to 73.2% recall, as compared to 49.6% recall reported in [8]. Figure 4.4 shows 2 of our retreived objects.

We have tested our approach on the challenging Oxford 5k dataset, introduced in [38]. It contains 5062 high resolution images of various buildings in Oxford, obtained by querying for building names on Flickr. Since, labels given to images tend not to be very accurate, this dataset contains a lot of distractors. Ground truth is available for 11 of these buildings in the form of *Good*, *Ok*, and *Junk* images. Out of these *Good* and *Ok* images are considered true positives and the *Junk* images are considered as "don't care" samples.

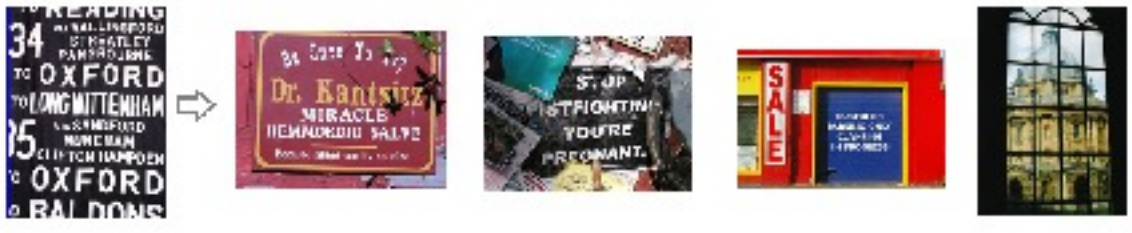



**Figure 4.5** Text is is the most common source of errors in our scheme. In this particular case, a text image got matched to the window structure in the final image, which contains the landmark Radcliffe Camera. Hence, these images also become a part of the cluster containing Radcliffe Camera and All Souls Building.

For Oxford Buildings dataset, the maximum amount of high order features which got extracted for an image was 45k. The size of the inverted index used was $2^{29}$. This gives a false positive rate of $8\times10^{-5}$ for indexing 45k elements. This implies a maximum of $3.6$ mismatches are expected to be generated between any pair of images. A total of 78 million high order features were extracted leading to a total memory requirement for indexing at $8 \times 512 + 4 \times 78 = 4408$Mb. The total time taken for extracting high order features and querying each image was $25 + 2 = 29$ minutes. For the high order features queried, the average length of the posting list was $1.16$.

We kept the match verification criteria as finding at least 3 primary features having at least 4 high order features each identical between a query and a database image. We have computed the clusters as the connected component on the graph of matching images. In all 317 clusters were discovered containing a total 1367 images in them. The largest cluster has 362 images showing nearby All Souls building and Radcliffe Camera from various viewpoints. We were also able to find many interesting smaller clusters. These results are shown in Figure 4.6. We got mismatches, mostly in the form of text images, as shown in Figure 4.5.

We tested the scalability of our approach using the Oxford 105K dataset used in [8]. A total of 1480 million high order features were extracted. We used an inverted index of size $2^{29}$, which resulted in an average length of posting list for the queried high order features at $6.8$. The effect of larger average length of posting list can be seen at the average query time per image, which increased from $0.024$ seconds to $0.086$ seconds. A faster average query time can be obtained by using a larger inverted index. The total time taken for extracting features and querying was $9 + 2.5 = 11.5$ hours. The memory requirement for indexing was $8 \times 512 + 4 \times 1480 = 10016$Mb. We used the verification criteria as finding at least 6 primary features having at least 4 high order features in common. We were able to obtain 2147 clusters involving $7198$ images, with the largest cluster having $2265$ images. Eyeballing this cluster revealed mismatches due to repeating patterns such as text, doors and windows which lead to coalescing of many smaller clusters.

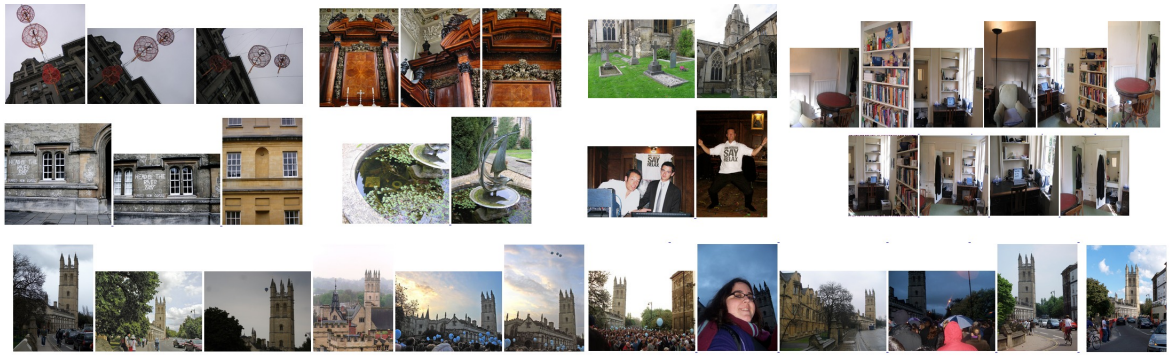


**Figure 4.6** Top two rows show small clusters identified by our method. Bottom row shows the cluster corresponding to 'difficult' Magdalen Tower.

## 4.4 Discussion

We show that it is feasible to index sufficient high order features capturing the appearance and geometric characteristics in an image, to an extent that there is no need for doing explicit geometrical verification. This is very advantageous as it eliminates any need for random disk accesses to fetch information required for doing geometrical verification. This is made possible as our geometric match verification criteria is computable directly from inverted index retrievals. We design an inverted indexing scheme which can adapt to the size of the database to ensure adequate sparsity of the posting lists to ensure constant time retrievals. The space savings are made by exploiting the behavior of an oversized Bloom filter using only one hash function. The extreme amounts of memory used by the Bloom filter is then shared efficiently using an inverted index structure for indexing multiple images. Our match verification is robust to the introduction of occasional spurious matches generated by our indexing scheme. Indexing high order features coming from an extensive domain would create problems for all the popular indexing schemes used for image retrieval or match graph construction, but we turn this to our advantage, by devising an indexing scheme based on Bloom filters, which uses constant storage per entry irrespective of its size or complexity.

In conclusion, our contributions can be summarized as: (i) introducing a novel indexing scheme suited for indexing and querying the geometrical and appearance information in images (ii) implementing an exhaustive pairwise matching scheme to build an image match graph for moderately large datasets in linear time (iii) introducing a geometric verification criteria verifiable at index.

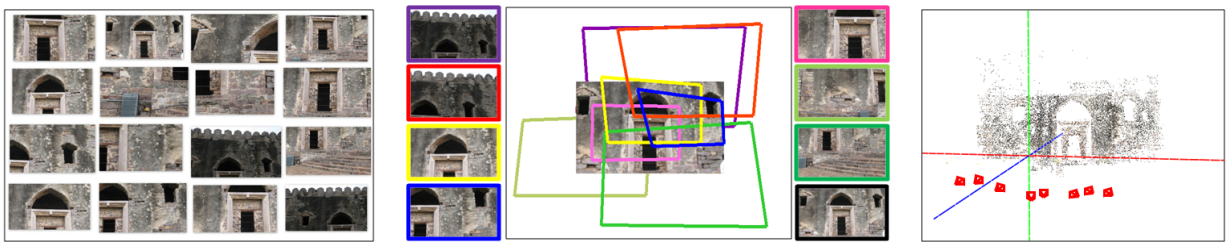
*Chapter 5*

# Creating Walkthroughs from Image Collections

The increasing popularity of digital photography and online photo-sharing sites such as Flickr is creating photo collections of landmarks and popular destinations around the world that are growing by the day. These massive datasets are visually interesting as they often capture a landmark site from a variety of viewpoints and in different illuminations and compositions. However, browsing such a massive unstructured photo collection can be difficult without any cues that indicate the relationship between the images. This has motivated a variety of approaches that try to organize photographs using geographical data, annotations, tags, etc. [42, 52, 23]. Attempts were also made to provide interactive and intuitive means of exploring photos and videos. The World-Wide Media Exchange (WWMX) [59] arranged images on an interactive 2D map, PhotoCompas [35] clustered images based on time and location, Realityflythrough [28] explored video from camcorders instrumented with GPS and tilt sensors, and Kadobayashi and Tanaka [21] presented an interface for retrieving images using proximity to a virtual camera. Image-based walkthroughs which worked on the principles of image based rendering and virtual view synthesis had also been created but from controlled acquisition of imagery [3]. Aspen Movie Map allowed a user to take a virtual tour of the city of Aspen, Colarado by registering images captured from a moving car onto an interactive street map of the city. Google Street View and EveryScape provide panoramic views from various positions along many streets in the world. In Photowalker [58], a user can manually author a walkthrough of a scene by specifying transitions between pairs of images in a collection. In these systems, location is obtained from GPS or is manually specified. Johansson and Cipolla [19] developed a system where a user can take a photograph, upload it to a server where it is compared to an image database to receive location information.

Recent advances in computer vision in robustly solving image matching and the recovery of 3D structure and camera pose from images via *structure from motion* (Sfm), was exploited by the system dubbed *Photo-Tourism*[52]. It created extremely effective virtual 3D walkthroughs of a scene from unstructured Internet photo collections of popular tourist locations. The image correspondences and 3D camera poses recovered automatically by this system[52], made it possible for users to interactively navigate images registered in 3D. However, the underlying pipeline did not scale to large photo collections. Running times reported in[54] varied from 11 hours for a 1K image dataset to $>$ 50 days for 8000

images. The primary computational bottlenecks were in the pairwise image matching step and the subsequent incremental 3D reconstruction stage where multiple rounds of global non-linear optimization referred to as *bundle adjustment* are performed. This is important since one of the overall goal in this system is to recover a single globally consistent reconstruction of the scene and all the cameras.



(a) (b) (c)

**Figure 5.1** Our interface for browsing image collections using walkthroughs (a) An input image collection (b) Our interactive image navigation interface. (c) One of the multiple partial reconstructions of the scene, computed from the images shown in (b).

We present a new system that generates interactive navigation of a photo-collection similar to Phototourism[52]. Our system however does not require a global 3D reconstruction of the scene and all the cameras. Rather, it relies only on partial local reconstructions which are independently estimated from subsets of nearby overlapping images and thereby allowing it to scale to large datasets.

In an image-based walkthrough, users primarily observe images or transitions between image pairs at any time. Rendering a realistic transition between an image pair via image based rendering techniques [46] requires the knowledge of relative pose between the corresponding cameras, a sparse 3D reconstruction of points observed by these two cameras and optionally a geometric reconstruction of the scene (required by advanced IBR techniques such as [46]). Therefore, a global Sfm reconstruction would allow direct transitions between any image pairs, as the relative pose of all pairs can be obtained from the global reconstruction. However, in practice only transitions between images whose views overlap tend to be the most useful. Our system limits the possible images that one would view next to a small set of proximal images whose views overlap with the current image. This is determined by the size of the subset for which a partial reconstruction is estimated.

Our approach for detecting these image subsets builds upon a state of the art image based retrieval technique [47, 37, 14, 40] which can efficiently retrieve duplicates or similar images based on visual appearance. This approach can be made scalable and more efficient by the use of inverted indices which map individual visual words to a list of images in which they occur. This can be very useful because generally only a small percentage of the visual words are present in an image. Each of the subsets are processed through a standard structure from motion pipeline. Restricting the size of the image subsets and processing them independent of each other reduces computation time and also makes it possible to

exploit parallelism during the reconstruction stage. We demonstrate the ability to scale to large datasets without sacrificing much on the user's navigation experience.

The pairwise image matching bottleneck is addressed in a novel way by the work of [23], where *iconic* images are first recovered using a global scene descriptor (Gist features) for clustering the images into small collections. Then the expensive pairwise matching is applied within these small clusters to find the iconic image of each cluster as the image having the greatest number of features in common with rest of the images. Each of the iconic images are then verified with respect to its top $n$ matches among the iconic images recovered using a visual word vocabulary based search. Another recent approach [42] avoids a globally consistent reconstruction of the scene in the context of robotic navigation to do scalable localization and mapping (to enable appearance based navigation) of the scene.

One of the key assumptions made by most previous systems is that all images will be available prior to processing, hence they are designed to process all the images in batch mode. However, online photo collections of important landmarks are often growing continuously and this indicates the need for an efficient online system that can incrementally insert new photographs into an existing reconstruction as they become available. Our incremental reconstruction framework maintains the photographs as a graph whose topology changes dynamically as new photographs become available. When a new image is successfully matched to existing photos in a pre-computed dataset, a new partial reconstruction is potentially added.

Our system is mostly immune to the various difficulties in computing a full global reconstruction via an incremental Sfm reconstruction approach, in particular its sensitivity to the choice of the initial image pair. We solve independent local Sfm problems and relax the need for global consistency in our reconstructions and camera poses. Thus, we avoid the catastrophic errors that occur when inaccuracies in camera pose estimation propagate and get compounded further along the sequence as new images, whose views overlap with the camera with erroneous pose, get added. This can lead to severe errors in large sections of a reconstruction [52, 54].

## 5.1 System Overview

Our system takes a set, $S$, of uncalibrated images. The overall system can be broken up into stages in which operations are performed on this set of images. The relevant information required in the next stage, or for the incremental addition of images, is stored in the form of a graph at each stage. The following is the summary of the various stages in our system:

- **Obtaining Putative matches:** Use Vocabulary based techniques, for every image $i \in S$, to efficiently and scalably find the set of neighbours, $N_i$, as the images with similarity in appearance with image $i$. Create a directed graph of images, $G_1$, to store $N_i$ for each $i$. Also store the histogram representation of the visual words present in $i$ in the node corresponding to image $i$ to allow easy comparison with a new incoming image during incremental insertion.

- **Geometric verification of the putative matches:** Estimate pairwise epipolar geometry to determine which images in $N_i$ were viewing a common 3D structure as image $i$, for every $i \in S$. The verified set of images are called verified neighbors of $i$, $V_i$. The information about the verified neighbours is stored in the form of a directed graph of images, $G_2$.

- **Calibrating each image with respect to its Verified neighbors:** Use a Sfm system on the set $i \cup V_i$, for every $i \in S$, to find the parameters required for displaying images and making a transition from one image to another in a virtual setting while navigating though the scene. Store this information in the form of a directed graph of images, $G_3$, where every edge stores the parameters required for making a transition.

- **Creating mirrored edges to improve connectivity:** For every pair of images, add an edge from $i$ to $j$, if an edge from $j$ to $i$ is present in $G_3$ and mark it as a mirrored edge. Store this new graph as $G_4$. While browsing the photo collection, the mirrored edges are simulated using their corresponding true edges due to which they were created. The set of images to which an edge go from the node corresponding to image $i$ is called as Registered neighbors of $i$, $R_i$.

- **Addition of new images:** Repeat the previous four steps for the new image while updating the corresponding graphs at each stage. Also add the new image to the set $S$ to allow matching with images coming in future.

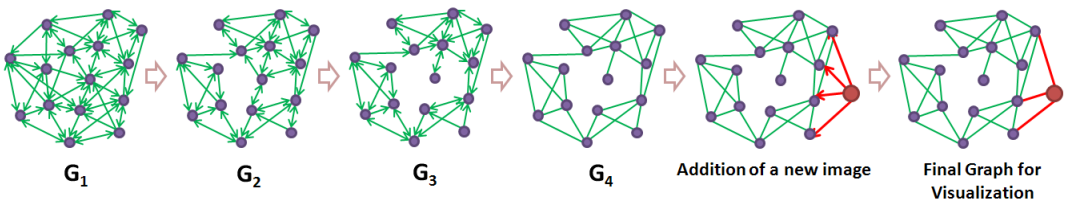



**Figure 5.2** Overview of our system for computing image based walkthroughs, highlighting the process of insertion of a new image

The distribution of vertex degrees in graph $G_4$ indicates how connected it is. Even though the pairwise relations between the images are commutative, $G_1$, $G_2$ and $G_3$ are represented as directed graphs in order to bound the number of spatial verifications required and the number of images on which Sfm is performed for an image in one round. Provided the above two points are true, it is safe to assume that under normal circumstances, a finite bound is put on the time taken in performing Sfm on the set of images $i \cup V_i$. This makes the time complexity of creating graph $G_3$ from $G_2$ approximately linear in the number of images and thus allowing it to scale to large number of images. Similarly, it is easy to see that creating $G_2$ from $G_1$ is also linear in the number of images in set $S$. Once a vocabulary is determined, then the representation of an image in the form a histogram of visual words depends upon

the number of visual words (which is bounded) and also on the number of features extracted from every image (which is also bounded). Therefore, we need to do an O($N^2$) matching of histograms of the images. These histograms are sparse in general. In such a scenario, the determination of the top matches becomes $O(N)$ with respect to the number of images under consideration by the use of inverted indices of visual words as in [40].

Our visualization scheme allows the user to navigate the edges and nodes present in the graph $G_4$ in a virtual 3D world. Our visualization scheme is similar to that of [52] in terms of using a single proxy plane for view interpolation, but better viewpoint interpolations can also be computed using 3D geometric proxies, [46]. When the user is on a node corresponding to an image $i$, we show $i$ in the center along with images in $R_i$ which are shown as wireframes oriented in space in a way so as to provide a cue about the relationship among the images in the set $i \cup R_i$.

In the following sections, we describe our system in more detail.

## 5.2 Image Matching

Our goal is to recover for each image $i$, a partial scene reconstruction based on the currently available images which were directly matched to $i$. The relative pose of all the cameras and a sparse set of 3D points reconstructed from these images are represented in a local coordinate system. Putative matches for any image can be easily computed by matching its features with the features extracted from the rest of the images and ranking the images on the basis of number of feature matched with the query image. Quantization of features, by the use of visual word vocabularies, can be employed to speed up the process of matching the features. This visual word vocabulary used for this can be created from the features of a representative set of images of the whole dataset. As a result, an image can now be concisely represented as a histogram of visual word frequency and thereby reducing the problem of comparing images to comparing histograms. Alternatively, a Vocabulary tree based image search used in [15] can also be employed to find the top matching images of a query image.

The Bag of visual word based model ignores the position of the features, and hence, some of the features may get incorrectly matched *based on appearance* across images. If two images are looking at the same portion of the scene, then the geometric model predicted by the feature matches across the images is expected to accurately predict the position of most of the matching features across the images. Thus, verification of a matching pair can be done by examining the number of correct predictions the best geometric model, estimated from the feature matches, is able to do. Alternatively, these false matching images can be identified and removed by the Sfm system, i.e. creating $G_3$ directly from $G_1$, but we prefer to run the Sfm system only on the verified matches of an image as indicated in $G_2$. This saves time in case the Sfm system does an exhaustive pairwise matching of images. Additionally, it saves time for the bundle-adjustment step typically employed by Sfm systems to refine the estimates and further reject non matching images. This pre-verification can also be used in the identification of a good initial pair with respect to image $i$ prior to applying Sfm on the set $i \cup V_i$. Specifying the initial pair

for reconstruction as such increases the chance of image $i$ getting registered by the Sfm system when it is run on Image $i$ and its neighbours. The following subsections give the implementation details:

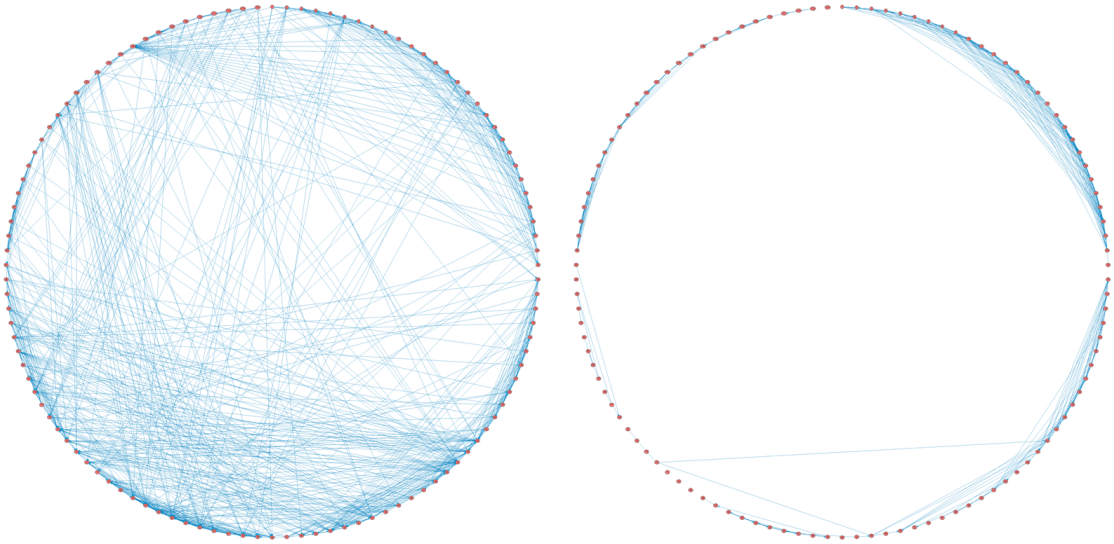


**Figure 5.3** HILLTOP dataset: Graphs showing putative matches and verified matches. Images are represented as dots on the circle. Edges represent match between two images. [left]Graph showing the matches obtained by Bag-of-words based matching. [right] Graph showing the matches obtained by the spatial verification of matches obtained by the Bag-of-Words framework.

### 5.2.1 Obtaining Putative Matches

As a first step towards identifying the verified neighbours of every image in $S$, we try to find the images which are similar in appearance. For this we extract robust SIFT features from all the images to apply a visual word vocabulary based image matching system. We used a representative set of images of the landmark to create a context specific vocabulary. The SIFT features from these images were clustered using the Kmeans algorithm implemented on GPU. Alternatively, to avoid the overhead of creating a vocabulary, we also used the Vocabulary tree based image search used in [15] in some experiments. While creating the histograms of the images, we used Term Frequency(tf) to normalize the difference in the number of SIFT points extracted per image. We also used Inverse Document Frequency(idf) to downplay the importance of commonly occurring words. We measure the similarity between two images on the basis the similarity of the histogram of the two images measured using Cosine similarity distance function, used in many document retrieval techniques. The top 10 matches according to this score are recorded for each image in $G_1$. Image matches obtained by considering the top 10 matches on a 114 image HILLTOP dataset having 3 different landmarks are shown in Figure 5.3. Note that it is not inferable from this graph that 3 different landmark are present in this dataset.

### 5.2.2 Geometric Verification

We do a refinement of the top matches returned for each image by the previous step to remove spurious matches. To verify an image pair, we first estimate a Fundamental matrix using RANSAC which can best explain the epipolar geometry of the matching features between the images. The inliers with respect to this fundamental matrix are computed and if the number of inliers is greater than the threshold (40 in our case), then the match is accepted. These images can be sent per se to the Sfm system for Geometry computation where some of the images may fail to get registered. The success of Geometry computation step depends significantly upon the initial pair of matching images used for starting the calibration process. The initial pair of images should have a good number of matching features and also have a good baseline. To identify the suitable matching pair for the reconstruction, we score them on the ratio of area which is covered by the inliers in each of the images as compared to the total area of the images. The area covered by the inliers is computed as the area covered by the convex hull enclosing all the inliers. The matching image which has the highest score with respect the concerned image, along with the concerned image, is taken as one of images in the initial pair for the reconstruction. Verified image matches obtained by verifying the top 10 matches on a 114 image HILLTOP dataset having 3 different landmarks are shown in 5.3. Note that it is inferable from this graph that 3 different landmark are present in this dataset.

## 5.3 Generating Partial Reconstructions

After the spatial verification stage, we obtain a set of 2D correspondences within the set of images $i \cup V_i$. We now perform structure from motion on these images, to estimate a partial metric reconstruction of cameras and 3D points. We use BUNDLER [49], a freely available Sfm implementation that first generates an initialization for all cameras and points using an incremental seed and grow approach and then performs several rounds of bundle adjustment and outlier removal to refine the full camera calibration parameters and the sparse 3D points. Note that these camera pose estimates are with respect to a local coordinate frame, selected arbitrarily for each partial Sfm problem we solve. However, this is sufficient to recover the relative pose between camera $i$ and any of its immediate neighbors in $V_i$.

### 5.3.1 Improving Connectivity of the Graph

The partial reconstruction corresponding to each image $i$ is referred to as $P_i$. $P_i$ comprises of reconstructed cameras for the set of images $i \cup V_i$ and a set of 3D points visible in these images. The relationship of any image $i$, with its registered neighbors $R_i$ can be represented in the form of a directed graph, $G_3$, in which a directed edge is present from image $i$ to every image $j \in R_i$. Any such edge means that a transition from the source to the destination image is possible while navigating the scene. Also, an edge from $i$ to $j$ can potentially be used to create an edge from $j$ to $i$ if it is not present. This is

| Function | Key |
|---|---|
| Zoom in | , |
| Zoom out | . |
| Show/Hide wireframes | Right Click |
| Transition to a new image | Left Click |
| Move | w/s/a/d |
| Rotate | z/x |
| Display neighboring images | r |

**Table 5.1** Table showing the controls provided by our visualization interface.

done by using the $P_i$ as a proxy for $P_j$ by centering $P_i$ with respect to $j$ while showing a transition for $j$ to $i$. This makes the graph used for navigating the scene, $G_4$, undirected.

## 5.4 Incremental Addition of new images

Our framework makes it possible to incrementally add new images into our system. For the new image, SIFT features are extracted and quantized by using the visual word vocabulary. These visual words are queried against the inverted index storing the pre-existing images to determine potentially matching images of the new image. The new image is then also inserted into the inverted index structure to allow matching with more incoming images. Next, a spatial verification is applied to the top 10 matches. The verified matches along with the new image are provided to BUNDLER to obtain a partial reconstruction for the new image. In $G_4$, all the registered matches of the new image get connected to the new image via a undirected link, i.e. a transition can be made from both the directions.

## 5.5 Image-based rendering framework

Our scene navigation scheme is inspired from the one used in the Phototourism system [52]. Initially, we show a 3D world corresponding to a partial reconstruction, $P_i$, of one of the images, $i$. Images are displayed by using a best fit plane(computed using RANSAC) corresponding of the points visible in the image as a proxy surface for projection. Initially, the virtual camera is placed congruent to the camera parameters recovered for image $i$ and image $i$ is projected on it proxy surface. Images in $R_i$ are shown as wireframes of their corresponding projections on their proxy planes.

The user can point and click at any of the wireframes to move to the partial reconstruction corresponding the a new image, $j$. This is done by showing a transition within $P_i$ during which the virtual camera moves from the camera parameters corresponding to image $i$ to the camera parameters corresponding to image $j$. Note that the recovery of metric camera calibration allows the possibility of estimating dense depth maps from the images, thus making it possible to use advanced image-based rendering techniques such as [45] for generating better transitions between photographs. The transition

is accompanied by showing a fading in of image $j$ and corresponding fading out of image $i$. At the end of the transition, we show the partial reconstruction $P_j$ in the same way as described above. Thus, the user is able to navigate the scene using partial reconstructions. Table 5.5 shows a list of controls provided by our visualization interface.

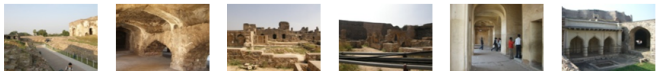
| Name | Sample Images | N | T |
|---|---|---|---|
| Hilltop |  | 114 | 42 minutes |
| Gate |  | 135 | 1 hour |
| Courtyard |  | 687 | 9 hours |
| Fort |  | 5989 | 124 hours |

**Figure 5.4** Descriptions of HILLTOP, GATE, COURTYARD and FORT datasets used in our experiments. N shows the number of images and T shows the time taken by our system.

## 5.6 Results

We have tested our system on a large collection of 6000 photographs of a heritage site, which we refer to as the FORT dataset. Some small subsets of this dataset which have been used in specific experiments are the 135 images of the GATE dataset, and the HILLTOP dataset containing 114 images. We also tested the performance of our incremental system on the COURTYARD dataset containing 687 images taken under different lighting conditions.

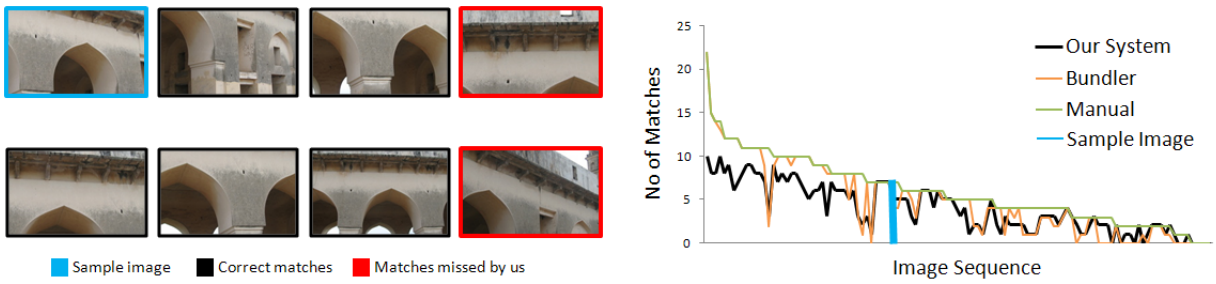



**Figure 5.5** HILLTOP dataset : [left]Matches determined by our system from $G_2$ for a sample image (shown in blue) compared with its matches in $G_m$; [right]Comparison of the number of matches with the manual graph by the two systems.

### 5.6.1 Experiments

We run the first 2 stages of the pipeline on the HILLTOP dataset to determine whether our system is able to robustly identify the correct neighbours of every image even when we are considering the top $n$ matches given by a non geometric test (recorded in $G_1$) which are further refined by a simple RANSAC based geometric test (recorded in $G_2$). The vocabulary used for creating $G_1$ was created using a set of 1088 randomly sampled images of the whole monument. We created an undirected match graph($G_m$), to be used as a ground truth, by manually comparing every image to every other image in the dataset. We also computed a match graph($G_b$), in a manner similar to [52] by running BUNDLER [49] individually on the 3 different clusters present in the dataset. Figure 5.5[right] reports the number of matches of each image obtained from $G_2$ and $G_b$ when compared against the set of neighbours obtained from $G_m$. Figure 5.5[left] shows an example image along with its matches obtained from $G_2$ compared with matches obtained from $G_b$.

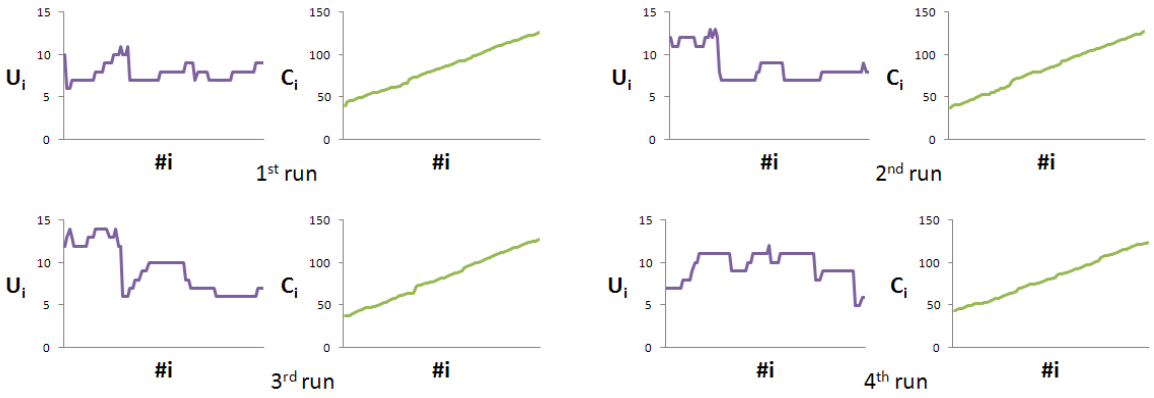



**Figure 5.6** GATE dataset : $U_i$ represents the number of unregistered images and $C_i$ represents the size of largest cluster for a set with $i$ images; the figure shows that $U_i$ converge to a small value and $C_i$ grows continuously indicating that more and more images get registered to form a single cluster and the number of unregistered images decrease

In another experiment, we simulated a scenario in which we initialize with a small set of random images, and images arrive over time. For this we generate random permutations of the images in the GATE dataset. We initialize our system from the first 50 images. We use the pre-trained vocabulary tree generated from large number of images from the internet which was made available by [14] for computing $G_1$ and incrementally add the rest of the 85 images. This was repeated for other permutations. The images which do not have any neighbour in $G_4$ are marked as unregistered images and the size of the largest cluster is also shown (Figure 5.6) with the addition of each image. The graph $G_4$ is very fragmented in the beginning, but as more images get added, smaller disconnected components in the match graph get merged resulting in a largest cluster size of 126, 127, 128 and 124 images respectively for the four runs. The final match graph is well connected and provides a pleasant navigation experience.

In a similar experiment with the COURTYARD dataset, we were able to get a cluster of 674 images out of 687 images while initializing from 200 images. This shows the applicability of our system on datasets of various sizes.

| No of Images | Time taken by our system | | | | Time taken by Bundler | | |
|---|---|---|---|---|---|---|---|
| | $G_1$ | $G_2$ | $G_3$ | Total | Matching | Bundle Adjustment | Total |
| 55 | 22 | 401 | 541 | **964** | 719 | 310 | **1029** |
| 65 | 27 | 474 | 635 | **1136** | 1092 | 409 | **1501** |
| 75 | 31 | 563 | 738 | **1332** | 1427 | 563 | **1990** |
| 85 | 35 | 649 | 858 | **1542** | 1858 | 660 | **2518** |
| 95 | 39 | 785 | 1165 | **1989** | 2437 | 1023 | **3460** |
| 105 | 43 | 847 | 1437 | **2327** | 2914 | 1183 | **4097** |
| 115 | 47 | 975 | 1768 | **2790** | 3621 | 1470 | **5091** |
| 125 | 51 | 1079 | 2112 | **3242** | 4459 | 2146 | **6605** |
| 135 | 55 | 1164 | 2382 | **3601** | 4955 | 2487 | **7442** |

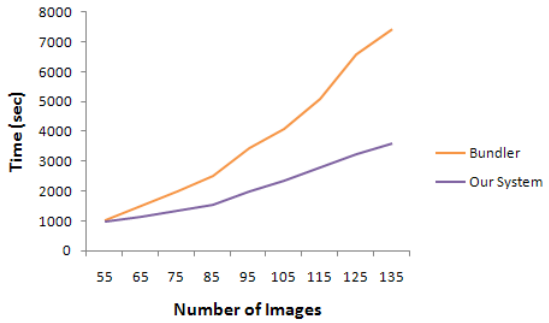


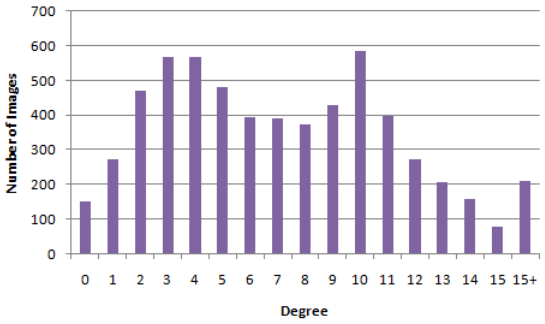

**Figure 5.7** GATE dataset: [left]Time taken in various stages of creating a walkthrough using our system as compared to time required to do a global reconstruction using BUNDLER [49] on the same set; [right] Graph comparing total time taken by the two systems.

In another experiment with the GATE dataset, we compare the time complexity of creating a walkthrough using our system to that of time taken to run the Sfm system on the whole dataset. We tested with sets of sizes 55 to 135 images with increments of 10 images. We use the pre-trained vocabulary tree made available by [14] for computing $G_1$. Time taken by our system is reported as the time required for matching ($G_1$), geometric verification ($G_2$) and creating partial scene reconstruction ($G_3$) for each image. The results are shown in 5.7[left].

In another experiment we demonstrate the scalability of our system on FORT dataset. We computed the graph $G_1$ of the set using the scene specific vocabulary created using a representative set of 1088 images from the FORT dataset. We report the degree of each image in 5.8[left]. The average degree obtained per node is 7.1 for the registered images in graph $G_4$ even when we consider only 10 matches per image in $G_1$. The largest connected component we obtain is of 4249 images and another cluster of 453 images. Thus, we demonstrate that a good navigation experience can be built in a scalable fashion from our system as we are able to obtain clusters of decent size with good connectivity on large datasets.

## 5.7 Discussion

We identified pairwise image matching and incremental 3D reconstruction involving bundle adjustment, used in Photo-tourism systems, as the bottlenecks for scaling to large image collections. Doing this is essential as this system requires a globally consistent 3D reconstruction of the whole image collection. We approach this problem from the users perspective to find that during visualization, the images near to the current image are the most useful. Therefore, our approach first identifies a limited




| Size of connected component | Number of Instances |
|---|---|
| 1 - 10 | 277 |
| 11 - 20 | 17 |
| 21 - 30 | 5 |
| 31 - 100 | 2 |
| 101 - 500 | 2 |
| 501 - 1000 | 0 |
| 1000+ | 1 |

**Figure 5.8** FORT dataset: [left]Histogram showing the number of images of each degree in graph $G_4$ for the FORT dataset; [right] Table showing number of connected components of various sizes present in $G_4$ after running our system on the FORT dataset

number these useful nearby images for every image using image retrieval techniques involving Bag of Words representation and spatial verification. We compute 3D reconstructions on these small sets to obtain a reconstruction of the neighborhood of every image. If computing a full 3D reconstruction for the whole image collection is seen as solving a large difficult problem, then our approach, in this analogy, can be described as solving several small difficult problems. Given only a definite maximum number of images are reconstructed as neighbors for every image, it is safe to assume a finite bound on computational cost. This makes our whole framework approximately linear in the number of images. Our visualization scheme is also simple as it only requires migration to the local reconstruction of the neighbouring image selected by the user.

## 5.8 Summary

We demonstrated that it is possible to achieve an image based browsing experience comparable to the one generated by a full reconstruction even by employing several partial reconstructions of a scene. The proposed approach is incremental, approximately linear in the number of images and massively parallel at every stage, and hence easily scalable to very large image collections. The ability to incrementally grow our reconstruction makes it well suited for browsing the ever growing photo collections.

*Chapter 6*

# Conclusions and Future Work

The rapid advances in digital technology has now made it possible to acquire, share and download large number of images. The automation in downloading has allowed anyone to aquire thousands of images of a particular topic. In this thesis, we have desribed our contributions to the ongoing efforts in the computer vision community to acquire meaningful information from these collections. In particular, we have presented techniques for organization and visualization of these image collections keeping in mind buildings and heritage sites.

Our first contribution is for the organization of these image collections as clusters of matching images. These clusters are easily obtainable if a image match graph is constructed showing images as nodes and a match between images as an edge. For this, we considered a novel problem of doing exhaustive pairwise matching of all the images in a scalable manner. This problem is difficult given the quadratic nature of the complexity of the problem. Our main inspiration for bringing scalablity to this problem comes from a well established baseline implementation of image retrieval which uses the Bag of visual words framework. This framework gains scalability by building an index over the quantized regions, also known as visual words, of the feature space of the features of database images. This index basically tells for a given visual word in the query image, which all images in the database contain this particular visual word. Hence, the average number of matches to be considered in the database images is proportional to the average size of the posting lists in the index which depends upon the number of visual words. The main limitation with this technique is that the number of visual words cannot be increased arbitrarily.

Our solution for doing exhaustive pairwise matching involves building an index similar to Bag of visual words framework, and querying every image in the collection to obtain its matches. In order to get scalability, we focus on obtaining a large enough feature space over which a sufficiently large index can be created. We obtain this features space by extracting high order features in images by combining visual words of nearby image features with the quantization of their relative affine geometry. This discreet feature space is large enough to allow us to reproject, using hash functions, into a sufficiently large number of bins, over which an inverted index is built subsequently. In particular, we choose the number of bins to be in proportion to the size of the database. This makes the average posting list size

constant which consquently implies contant query time for querying one image, making querying all the images feasible.

We choose these bins as the bit positions of Bloom filters, which are set membership query datastructure designed for trading off retrieval accuracy for space efficieny. As a result, querying over this index indirectly means querying the respective Bloom filters of each of the database images in the index. In order to minimize computation required we use only one hash fuction in these Bloom filters. A potential drawback in this setting is the increase in chance of getting false postives while querying getting. This effect is mitigated by the fact that very large Bloom filters are implied by the large inverted index we used for indexing. Moreover, our spatial verification criteria robust to occasional false correspondence detections. Our approach also does not require a post verification step as our spatial verification criteria is computable directly from index retrievals, enable us to do exhaustive pairwise matching in linear time complexity. We validated our approach on a moderately large dataset of 5k and 100K images. Our approach was also effective in finding small clusters of matching images in the University of Kentucky benchmark dataset.

Our second contribution is designing a scalable and incremental framework to create walkthroughs in large image collections. This is achieved by bypassing the scalability and error propagation issues faced by a benchmark technique employing structure from motion to do global reconstruction using all the images in a incremental manner. Our approach involves determinig a fixed number of top neighbours of each image using standard bag of words model based image retrieval followed by reranking with geometric verification. A local reconstruction is created for each of the images in database using the top neighbours. This is much easier easier than doing a complete reconstruction of the scene which need to employ an expensive bundle adjustment procudure after each iterative addition of images in order to keep error accumulation in check. Our approach is shown is shown to be approximately linear in the number of images in the database.

Our browsing scheme shows one of these reconstructions, and upon a transition to neighbouring image, loads the reconstruction corresponding to the transitioned images. This gives the user an illusion of browsing a global reconstruction. We thus demonstrate that it is possible to achieve an image based browsing experience comparable to the one generated by a full reconstruction even by employing several partial reconstructions of a scene. The proposed approach is incremental, approximately linear in the number of images and massively parallel at every stage, and hence easily scalable to very large image collections. The ability to incrementally grow our reconstruction makes it well suited for browsing the ever growing photo collections. We validate our approach using a Golkonda Fort image dataset having around 6k images.

The digital revolution has openend gates to a deluge of data of all sorts. However, we are yet to witness the kind of advances made in the field textual data, in the visual domain. This is mainly because textual data is clean, segmented, one dimensional and easily indexable, and on the other hand visual data is noisy, unsegmented and not unidimensional. Hence, there is a lot of scope for improvement at these basic levels. However, we feel there is an often neglected leeway while mining for matching images,

which is in matching images being likely to be discovered reliably even when a few of the matching features are mismatched, given the profusion of features extracted per image. We translated this leeway into a novel indexing scheme which trades accuracy for computational speed and memory efficiency. In future, we hope to extend this idea to seeming related frameworks such as sparse dictionaries, cortical learning algorithms, finding nearest neighbours etc. We are looking in the direction of extracting more repeatable and more robust high order features: the high order features we currently use encode affine geometry of a pair of features which is good enough only for features lying on a plane. Our other contribution for visualizing image collection builds upon the observation that which navigating, users are mostly interested in nearby images. We exploited this observation to apply a divide and conquer paradigm to the problem of obtaining a reconstruction of the scene, while not requiring to recombine the solutions to the subproblems. We are searching for more complex problems which meet the criteria of not explicity requiring a recombining step. Finally, we believe that the two techiques are significant steps towards the big goal of organizing and visualizing visual data available on the internet.

# Related Publications

- Kumar Srijan, Syed Ahsan Ishtiaque, Sudipta N. Sinha and C.V. Jawahar, “Image-based Walkthroughs from Incremental and Partial Scene Reconstructions”, in Proceedings of the British Machine Vision Conference(BMVC), 2010 at Aberystwyth, UK
- Kumar Srijan and C.V. Jawahar, “Towards Exhaustive Pairwise Matching in Large Image Collections”, in Proceedings of the Workshop on Web-scale Vision and Social Media, in conjuction with European Conference on Computer Vision(ECCV), 2012 at Firezine, Italy

# Bibliography


[1] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski. Bundle adjustment in the large. In *ECCV (2)*, pages 29–42, 2010.

[2] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *ICCV*, 2009.

[3] D. G. Aliaga, T. Funkhouser, D. Yanovsky, and I. Carlbom. Sea of images. *Visualization Conference, IEEE*, 2002.

[4] H. Bay, T. Tuytelaars, and L. J. V. Gool. Surf: Speeded up robust features. In *ECCV (1)*, pages 404–417, 2006.

[5] S. Bhattacherjee, A. Narang, and V. K. Garg. High throughput data redundancy removal algorithm with scalable performance. In *HiPEAC'11*, pages 87–96, 2011.

[6] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[7] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations (extended abstract). In *STOC*, 1998.

[8] O. Chum and J. Matas. Large-scale discovery of spatially related images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(2):371–377, 2010.

[9] O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *CVPR*, pages 17–24, 2009.

[10] D. J. Crandall, A. Owens, N. Snavely, and D. Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. In *CVPR*, pages 3001–3008, 2011.

[11] L. Fei-Fei. Tutorial on bag-of-visual-words. In *CVPR*. IEEE Computer Society, 2007.

[12] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[13] J.-M. Frahm, P. F. Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, and S. Lazebnik. Building rome on a cloudless day. In *ECCV (4)*, 2010.

[14] F. Fraundorfer, C. Engels, and D. Nistér. Topological mapping, localization and navigation using image collections. In *IROS*, pages 3872–3877, 2007.

[15] F. Fraundorfer, C. Wu, J.-M. Frahm, and M. Pollefeys. Visual word based location recognition in 3d models using distance augmented weighting. In *3DPVT*, 2008.

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

[17] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[18] K. Heath, N. Gelfand, M. Ovsjanikov, M. Aanjaneya, and L. J. Guibas. Image webs: Computing and exploiting connectivity in image collections. In *CVPR*, pages 3432–3439, 2010.

[19] B. Johansson and R. Cipolla. A system for automatic pose-estimation from a single image in a city scene. In *IASTED int. conf. Signal Processing, Pattern Recognition, and Applications*, 2002.

[20] T. Kadir, A. Zisserman, and M. Brady. An affine invariant salient region detector. In *ECCV (1)*, pages 228–241, 2004.

[21] R. Kadobayashi and K. Tanaka. 3d viewpoint-based photo search and information browsing. In *SIGIR*, pages 621–622, 2005.

[22] H. Li and R. I. Hartley. Five-point motion estimation made easy. In *ICPR (1)*, pages 630–633, 2006.

[23] X. Li, C. Wu, C. Zach, S. Lazebnik, and J.-M. Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In *ECCV (1)*, pages 427–440, 2008.

[24] Y. Li, N. Snavely, and D. P. Huttenlocher. Location recognition using prioritized feature matching. In *ECCV (2)*, pages 791–804, 2010.

[25] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999.

[26] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[27] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *BMVC*, 2002.

[28] N. J. McCurdy and W. G. Griswold. A systems architecture for ubiquitous video. In *MobiSys*, pages 1–14, 2005.

[29] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *In Proceedings of the 8th International Conference on Computer Vision*, pages 525–531, 2001.

[30] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *ECCV (1)*, pages 128–142, 2002.

[31] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[32] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, 2005.

[33] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. J. V. Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72, 2005.

[34] M. Mitzenmacher. Compressed bloom filters. *IEEE/ACM Trans. Netw.*, 10(5):604–612, 2002.

[35] M. Naaman, Y. J. Song, A. Paepcke, and H. Garcia-Molina. Automatic organization for digital photographs with geographic coordinates. In *JCDL*, pages 53–62, 2004.

[36] D. Nister. An efficient solution to the five-point relative pose problem, 2004.

[37] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR (2)*, pages 2161–2168, 2006.

[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 and A. Zisserman. Object mining using a matching graph on very large image collections. In *ICVGIP*, pages 738–745, 2008.

[40] J. Philbin and A. Zisserman. Object mining using a matching graph on very large image collections. In *ICVGIP*, pages 738–745, 2008.

[41] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "how do i organize my holiday snaps?". In *ECCV (1)*, pages 414–431, 2002.

[42] S. Segvic, A. Remazeilles, A. Diosi, and F. Chaumette. Large scale vision-based navigation without an accurate global reconstruction. In *CVPR*, 2007.

[43] I. Simon, N. Snavely, and S. M. Seitz. Scene summarization for online image collections. In *ICCV*, pages 1–8, 2007.

[44] I. Simon, N. Snavely, and S. M. Seitz. Scene summarization for online image collections. In *ICCV*, pages 1–8, 2007.

[45] S. N. Sinha, P. Mordohai, and M. Pollefeys. Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In *ICCV*, 2007.

[46] S. N. Sinha, D. Steedly, and R. Szeliski. Piecewise planar stereo for image-based rendering. In *ICCV*, 2009.

[47] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, pages 1470–1477, 2003.

[48] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, pages 1470–1477, 2003.

[49] N. Snavely. Bundler, 2007. http://phototour.cs.washington.edu/bundler/.

[50] N. Snavely. Scene reconstruction and visualization from internet photo collections, 2008.

[51] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski. Finding paths through the world's photos. *ACM Trans. Graph.*, 27(3), 2008.

[52] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846, 2006.

[53] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846, 2006.

[54] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2008.

[55] N. Snavely, S. M. Seitz, and R. Szeliski. Skeletal graphs for efficient structure from motion. In *CVPR*, 2008.

[56] K. Srijan, S. A. Ishtiaque, S. Sinha, and C. V. Jawahar. Image-based walkthroughs from incremental and partial scene reconstructions. In *BMVC*, 2010.

[57] H. Steinhaus. Sur la division de corps matriels en parties. In *Bull. Acad. Polon. Sci.*, 1956.

[58] H. Tanaka, M. Arikawa, and R. Shibasaki. A 3-d photo collage system for spatial navigations. In *Digital Cities*, pages 305–316, 2001.

[59] K. Toyama, R. Logan, and A. Roseway. Geographic location tags on digital images. In *ACM Multimedia*, pages 156–166, 2003.

[60] T. Tuytelaars and L. J. V. Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *BMVC*, 2000.

[61] X. Wang, M. Yang, T. Cour, S. Zhu, K. Yu, and T. X. Han. Contextual weighting for vocabulary tree based image retrieval. In *ICCV*, pages 209–216, 2011.

[62] Y. Xu and R. Madison. Robust object recognition using a cascade of geometric consistency filters. In *AIPR09*, pages 1–8, 2009.

[63] M. Zhang, M. C. Chan, and A. L. Ananda. Connectivity monitoring in wireless sensor networks. *Pervasive and Mobile Computing*, 6(1):112–127, 2010.

[64] Y. Zhang and T. Chen. Efficient kernels for identifying unbounded-order spatial features. In *CVPR*, pages 1762–1769, 2009.

[65] Y. Zhang, Z. Jia, and T. Chen. Image retrieval with geometry-preserving visual phrases. In *CVPR*, pages 809–816, 2011.