# Mining Characteristic Patterns from Visual Data

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

MS by Research
*in*
Computer Science

by

Abhinav Goel
200702001
abhinavgoel@students.iiit.ac.in

CENTER FOR VISUAL INFORMATION AND TECHNOLOGY
International Institute of Information Technology
Hyderabad - 500 032, INDIA
DECEMBER 2012

International Institute of Information Technology
Hyderabad, India

# CERTIFICATE

It is certified that the work contained in this thesis, titled " Mining Characteristic Patterns From Visual Data " by Mr. Abhinav Goel, has been carried out under my supervision and is not submitted elsewhere for a degree.

| | |
|---|---|
| Date | Adviser: Prof. C. V. Jawahar |

To My Family.

# Acknowledgments

In a meeting with my mother, my research advisor, Prof. C. V. Jawahar emphasized on the importance of doings things in the best manner that they can be done, and not merely doing them for the sake of it. Today, I realize the importance of that statement, because when I look back at the past one and a half years of my life, I feel proud to be the author of the research work in my thesis. First of all I'd like to thank him, for he has been my guiding lamp in both personal as well as professional endeavors in course of my research. Without his guidance, persuasion, and his ability to bring out the best in people, this thesis would not have been possible.

I am fortunate to have such wonderful friends in my colleagues at CVIT. They have been with me through thick and thin, always lighting up the mood with their witty jokes whenever I was tensed - Vinay, Mayank Juneja, Siddhartha, Yashaswi, Siddharth Khereda, Shrikant Baronia, Rohit Nigam, Rohit Gautam, Harshit Sureka, Shashank Mujjumdar, Jay Guru, Ankit, Chandrashekhar, Harshit Agarwal, Raman, Srijan, Nisarg, Nagender, Aayush, and Anand. I am grateful to Mr. R. S. Satyanarayana whose administrative support was indispensable at times. I would like to thank Phani whose presence at CVIT has helped students in more ways than one. Many thanks to Prof. P. J. Narayan, Prof. Jayanthi and Prof. Anoop for their encouraging presence and for providing an environment conducive to learning of the finest quality at CVIT.

I would like to thank Ankita, a dearest friend, without whom I wouldn't have made it so far. She has been my friend when I was alone, my guide when I was lost, and my support when I was about to give up. I would also like to thank my friends Ammar, Rakshit, Rahul, Ankur, Sankalp for making these 1.5 years one of the best years of my college life.

Finally, I would like to acknowledge the support of my family in my academic and research pursuits. Their belief in me is much stronger than my belief in myself, which has goaded me to push my limits - especially my mother and father, who were there with me in the darkest times.

Many thanks to everyone else who affected my life in any way, and wasnt acknowledged personally above.

# Abstract

Recent years has seen the emergence of thousands of photo sharing websites on the Internet where billions of photos are being uploaded every day. All this visual content boasts a large amount of information about people, objects and events all around the globe. It is a treasure trove of useful information and is readily available at the click of a button. At the same time, significant effort has been made in the field of text mining, giving birth to powerful algorithms to extract meaningful information scalable to large datasets. This thesis leverages the strengths of text mining methods to solve real world computer vision problems. Leveraging such techniques to interpret images is accompanied by its own set of challenges. The variability in feature representations of images makes it difficiult to match images of the same object. Also, there is no prior knowledge about the position or scale of the objects that have to be mined from the image. Hence, there are infinite candidate windows which have to be searched.

The work at hand tackles these challenges in three real world settings. We first present a method to identify the owner of a photo album taken off a social networking site. We consider this as a problem of prominent person mining. We introduce a new notion of prominent persons, where information about the location, appearance and social context is incorporated into the mining algorithm to be effectively able to mine the most prominent person. A greedy solution based on an eigenface representation is proposed and We mine prominent persons in a subset of dimensions in the eigenface space. We present excellent results on multiple datasets - both synthetic as well as real world datasets downloaded from the Internet.

We next explore the challenging problem of mining patterns from architectural categories. Our mining method avoids the large numbers of pair-wise comparisons by recasting the mining in a retrieval setting. Instance retrieval has emerged as a promising research area with buildings as the popular test subject. Given a query image or region, the objective is to find images in the database containing the same object or scene. There has been a recent surge in efforts in finding instances of the same building in challenging datasets such as the Oxford 5k dataset[46], Oxford 100k dataset and the Paris dataset[48]. We leverage the instance retrieval pipeline to solve multiple problems in computer vision. Firstly, we ascend one level higher from instance retrieval and pose the question: *Are Buildings Only Instances?* Buildings located in the same geographical region or constructed in a certain time period in history often follow a specific method of construction. These architectural styles are characterized by certain features which distinguish them from other styles of architecture. We explore, beyond the idea of buildings as instances, the possibility that buildings can be categorized based on the architectural style. Certain char-

acteristic features distinguish an architectural style from others. We perform experiments to evaluate how characteristic information obtained from low-level feature configurations can help in classification of buildings into architectural style categories. Encouraged by our observations, we mine characteristic features with semantic utility for different architectural styles from our dataset of European monuments. These mined features are of various scales, and provide an insight into what makes a particular architectural style category distinct. The utility of the mined characteristics is verified from Wikipedia.

We finally generalize the mining framework into an efficient mining scheme applicable to a wider varieties of object categories. Often the location and spatial extent of an object in an image is unknown. The matches between objects belonging to the same category are also approximate. Mining objects in such a setting is hard. Recent methods [55] model this problem as learning a separate classifier for each category. This is computationally expensive since a large number of classifiers are required to be trained and evaluated, before one can mine a concise set of meaningful objects. On the other hand, fast and efficient solutions have been proposed for the retrieval of instances (same object) from large databases. We borrow, from instance retrieval pipeline, its strengths and adapt it to speed up category mining. For this, we explore objects which are "near-instances". We mine several near-instance object categories from images obtained from Google Street View. Using an instance retrieval based solution, we are able to mine certain categories of near-instance objects much faster than an Exemplar SVM based solution.

# Contents

# List of Figures

# List of Tables

*Chapter 1*

# Introduction

With the advent of the Internet, a large amount of data is readily available and easily accessible. This data can be in the form of text, photos, music, and videos. For the regular Internet user, all this information is but a big black box. However, from an engineer's point of view, this data is a treasure trove of useful information, which can be tapped into and used for a variety of purposes. Several business decisions in major organizations are already being made based on the inferences from the text data collected from various sources. Departmental stores use information about customer transactions to facilitate sales by modifying arrangement of items on shelves, increasing stock of items which give a larger profit, and introducing schemes which benefit regular and trusted customers of the store.

Algorithms in data mining are adept at obtaining useful information from a large amount of data. An immediate application of such algorithms is to the problems of Computer Vision. In Computer Vision, we wish to empower a machine to observe and understand its surroundings with the effectiveness and speed with which a human can. Humans are intelligent beings. We are, with almost $100\,\%$ accuracy, able to identify any object (seen before) from afar, in the dark, or even when it is only partially visible. Each one of us has his/her own understanding of the social network of people around us. Just by observation, without interacting with many of the people, we can precisely say who is friends with whom, what are the current fashion trends, and what are the most pursued interests of people. With advances in the areas of both networking as well as vision independently, there is an increasing need to make existing vision algorithms to be able to process millions of millions of bytes of data obtained from the Internet. This is where data mining and vision cross paths. While mining text data is a well-established research area, still a lot needs to be done when mining multimedia. Working with images and video poses greater challenges in terms of computation time as well as storage space.

The objective of our work is to explore the applications of mining from large image databases. We investigate various possibilities where Computer Vision can benefit from the efficiency and speed that is Mining. Rather than restricting ourselves to a specific class of mining algorithms, we focus on mining interesting patterns from images for multiple tasks, using different methods. We discuss the various challenges associated with employing efficient mining techniques, and how they can be overcome.

## 1.1 Problem Setting

The application of mining to solve computer vision problems is not straightforward and is accompanied by its own set of challenges.

**Variability in Feature representations:** While entities in text data are well-defined and discrete, the same cannot be said for image features. An image feature is the basic unit for representing visual data. For example, a person is represented by his/her name, which is unique and consistent across all records. Now consider an image database, where each person is represented using a face image. There are bound to be minor variations across images of the face of the same person, which will lead to variability in image features used to represent those faces.

**Unsupervised setting:** While mining from visual data, there is no prior knowledge about the appearance or location of the object which we are trying to mine. Each image contains millions of windows of various sizes. An exhaustive evaluation of all windows to filter out useful information is a challenging problem.

**Scalability and Efficiency:** Each image yields a large number of features. The mining techniques to be applied to solve computer vision problems should be adept at working with billions of image features and at the same time be capable of processing and giving results in a reasonable amount of time.

**What are interesting patterns?** Interestingness of mined patterns can be perceived in multiple ways. The notion of what is interesting is varied. One person might be interested in the most frequently occurring objects in a movie scene. Another person might be more interested in how often a particular person or object appears in the movie, no matter how often it is visible. Building on top of the uncertainty in feature representations, we should be able to correctly identify the patterns that we deem interesting.

We address the problem of mining characteristic patterns from visual data, accommodating the uncertainty in representation of images as feature vectors. Our data constitutes real world databases manually collected from the Internet. We mine a variety of patterns, whose utility depends on the database being employed as well as the mining method. Various techniques to compensate for the uncertainty in feature representations and obtain useful results have been proposed.

## 1.2 Focus of this Thesis

There have been several attempts at applying well-established methods borrowed from the data mining field to solve computer vision problems. Some of the popular Computer Vision tasks where mining has been successfully applied are image classification, object detection, image retrieval, object category discovery and many others. Researchers have been able to find solutions to overcome the inherent issues in adapting standard data mining techniques to computer vision tasks. In this thesis, we have explored how mining can be adapted for identifying characteristic patterns from image data. Rather than' restricting ourselves to mining frequent item sets, we look at mining from a broader perspective, and its applications. Our contributions are threefold:-

**Mining Prominent Persons from Photo Albums** There are thousands of photo sharing websites on the Internet, with millions of photographs uploaded by a large user base. The photos which a person uploads tell a lot about his/her lifestyle, the people he/she usually hangs out with, and the places he/she visits. All these photos are readily available and boast a large amount of information about people, objects and events all around the globe. Identifying each of these is a unique Computer Vision problem in itself. Our objective is to solve a subset of these problems, which is that of mining the most prominent person in a given set of photo albums.

**Mining Characteristic Patterns for Architectural Style Categories** Buildings have long been a popular subject of instance retrieval problem in Computer Vision - Given a bounding box around a particular building, the task is to retrieve all instances of the same building from a database of images. The challenge is to work across variation in location, scale, and viewpoint. We move beyond the idea of buildings as instances, and explore the possibility of building as categories of architectural styles. To solve this problem, we demonstrate the utility of pairs of features mined for each category of architectural style by improving recognition of different buildings of the same architectural style. We create our own dataset of images of various European monuments downloaded from Google Image Search and Flickr[1]. We further propose a method to automatically discover features which are characteristic of a particular architecture, and help in distinguishing it from monuments of other styles.

**Leveraging Instance Retrieval for Efficient Category Mining** The problem of retrieval of objects belonging to the same category has long been studied. The popular method is to train a classifier for a set of object exemplars, and then search through the millions of possible windows (generated from few thousand images) and recognize objects. This process, despite giving reasonably good results, involves large amount of computation, making it unfeasible for large number of categories and/or large database size. On the other hand, instance retrieval solves a similar problem, albeit searching for the same object rather than different objects of the same category, in the order of milliseconds. We leverage the standard instance retrieval problem to solve the problem of category retrieval on a characteristic set of object categories. We further propose a mining method for automatically discovering such object categories from an image database.

## 1.3 Organization of the Thesis

The organization of this thesis is as follows. In Chapter 2, we give a detailed review of the previous works which solve various Computer Vision tasks using mining methods. We also give background knowledge about various areas in data mining as well as computer vision which form the basis for the work proposed in this thesis. In Chapter 3 we talk about our method of mining the most prominent person in a given photo album. We perform exhaustive experiments on various synthetic and real-world data and show results. In Chapter 4 we make a transition to architectural images obtained from Flickr. We propose a method to automatically discover characteristics which distinguish a particular

[1] http://flickr.com

architectural style from other styles of architecture. We show how characteristic low-level features can be utilized to improve classification and retrieval tasks. Generalizing to object categories, we leverage the powerful instance-retrieval pipeline to solve the category retrieval problem in Chapter 5. We mine various "near-instance" object categories from images obtained from Google Street View. We conclude in Chapter 6 and discuss the possible future research directions which begin where this thesis ends.

*Chapter 2*

# Background and Related Work

## 2.1 Data Mining

In this chapter, we talk about the popular techniques used for mining from text as well as visual data. We start with a discourse on various unsupervised mining techniques for mining interesting patterns. We then move on to talk about the prominent methods for image representation and retrieval. All these techniques form the building blocks of the various mining approaches proposed in this thesis.

### 2.1.1 Problem Formulation

The motivation for mining frequent itemsets from a transaction database stems from market basket analysis. The typical business decisions a retailer has to make from market basket data include identifying items that maximize profit and arrangement of items on the shelves in the supermarket. Consider a set of items brought by a few customers on any particular day at a general store (Figure 2.1). Each transaction is denoted by a transaction identifier $t_i$. Bread is brought in 4 out of the 5 transactions on that particular day, making its frequency 4/5. Similarly, butter is brought in 3 out of the 5 transactions, making its frequency equal to 3/5. However, we are interested in identifying frequent sets of items. For a given transaction database containing $N$ unique items, there are total of $2^N$ possible itemsets which need to be enumerated and evaluated. For the given example (Figure 2.1), there are 5 distinct items (bread, jam, butter, milk and beer). The total number of possible itemsets is $2^5 = 32$. This number increases exponentially with increase in number of distinct items. In this section we formalize the problem of mining frequent itemsets, and the popular algorithms which have been used to solve it.

### 2.1.2 Frequent Itemset Mining

Let $I = \{I_1 \ldots I_p\}$ be a set of $p$ items. Let $D$, the task-relevant data, be a set of database transactions where each transaction $T$ is a set of items such that $T \subseteq I$. Each transaction is associated with an identifier, called TID(T). A transaction contains $X$, a set of items in $I$, if $X \subseteq T$.

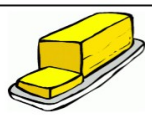
| Transaction | Items | | | | |
|---|---|---|---|---|---|
| T1 |  |  |  | | |
| T2 |  | |  | | |
| T3 |  | |  |  | |
| T4 |  | | | | |
| T5 | | | |  |  |

**Figure 2.1** Transaction database showing items bought in 5 transactions on a particular day at a departmental store.

A set of items is referred to as an itemset. An itemset that contains $k$ items is a $k-itemset$. The set $\{$bread, butter$\}$ is a 2-itemset. The occurence frequency of an itemset is the number of transactions that contain the itemset. This is also known, simply, as the frequency, support count, or count of the itemset. The support of an itemset $X \subseteq I$ is given by

$$support(X) = \frac{|\{T \in D | X \subseteq T\}|}{|D|}$$

If the support of the itemset $X$ satisfies the minimum support threshold $(support(X) > s)$, then it is said to be frequent, where $s$ is the minimum support threshold. Several methods have been proposed to mine frequent itemsets efficiently. The Apriori[4] algorithm is a classic algorithm to find the frequent itemsets in a given database for a given minimum support value.

### 2.1.3 The Apriori Algorithm

Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 [4] for mining frequent itemsets for Boolean association rules. The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset properties. It generates candidate itemsets of size $k+1$ from itemsets of length $k$ using an iterative level-wise search. First, the database is scanned to find the set of 1-frequent itemsets by accumulating the count for each item and storing those which satisfy the minimum support threshold. The resulting set is denoted $L_1$. Next $L_1$ is used to find $L_2$, the set of

2-frequent itemsets and so on. The number of times the database is scanned equals the length of the largest frequent pattern mined, or the maximum value of $k$ in $L_k$.

The Apriori property is used to improve the efficiency of the level-wise generation of frequent itemsets.

**Apriori Property:** All nonempty subsets of a frequent itemset must also be frequent.

The Apriori property is based on the following observation. By definition, if an itemset $X$ does not satisfy the minimum support threshold, $s$, then $X$ is not frequent. If an item $A$ is added to the itemset $X$, then the resulting itemset (i.e. $X \oplus A$) cannot occur more frequently than $X$. Hence, the count of an itemset $X$ is an upper bound on the count of all it's supersets. The Apriori algorithm follows a two-step process, consisting of **join** and **prune** actions.

1. **The join step:** To find $L_k$, a set of candidate $k-itemsets$ is generalized by joining $L_{k-1}$ with itself. This set of candidates is denoted by $C_k$. Let $l_1$ and $l_2$ be itemsets in $L_{k-1}$. The notation $l_i[j]$ refers to the $j^{th}$ item in $l_i$ (e.g., $l_1[k-1]$ refers to the second to the last item in $l_1$). For convenience, we assume that items with a transaction are sorted in lexicographic order. For the $(k-1)$-itemset, $l_i$, this means that items are sorted such that $l_i[1] < l_i[2] < \ldots < l_i[k-1]$. The join, $L_{k-1} \bowtie L_{k-1}$, is peformed, where members of $L_{k-1}$ are joinable if their first $(k-2)$ items are in common. That is, members $l_1$ and $l_2$ of $L_{k-1}$ are joined if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \ldots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] = l_2[k-1])$. The condition $l_1[k-1] < l_2[k-1]$ simply ensures that no duplicates are generated. The resulting itemset formed by joining $l_1$ and $l_2$ is $l_1, l_2, \ldots, l_1[k-2], l_1[k-1], l_2[k-1]$.

2. **The prune step:** $C_k$ is a superset of $L_k$, that is, it's members may or may not be frequent, but all of the frequent k-itemsets are included in $C_k$. A scan of the database to determine the count of each candidate in $C_k$ would result in the determination of $L_k$ (i.e. all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to $L_k$). $C_k$, however, can be huge, and so this could involve heavy computation. To reduce the size of $C_k$, the Apriori property is used as follows. Any $(k-1)-itemset$ that is not frequent cannot be a subset of a frequent $k-itemset$. Hence, if any $(k-1)-subset$ of a candidate $k-itemset$ is not in $L_{k-1}$, then the candidate cannot be frequent either and so can be removed from $C_k$.

## 2.2 Mining in Computer Vision

Frequent Itemset Mining has been applied to solve several problems in Computer Vision. Despite its effectiveness, it has still not been able to achieve state-of-the-art results on standard data sets. We give an insight into various ways in which data mining has benefited computer vision, and the tasks that have been solved. Creating the bag-of-items data for application of mining algorithm is an important step in mining. In most computer vision problems which employ frequent itemset mining techniques, images

are represented with a bag-of-words (BoW) method, i.e. histograms of vector quantized image features. Local features are preferred, since they result in sparse representation, better signal-to-noise ratio, and an increased robustness to image clutter. Several ways of defining a transaction from a bag-of-words image representation have been defined.

### 2.2.1 Clustering

A simple method of discovering characteristic (frequently occurring) patterns in data is to perform clustering. In clustering, similar objects are grouped into a single cluster. There are several popular algorithms which are employed for performing clustering on various types of data. For example, the K-means algorithm is the suitable choice of clustering method when creating visual vocabularies. In this section, we talk about the agglomerative clustering method which has been employed in our prominent person mining method.

#### 2.2.1.1 Hierarchical Clustering

Hierarchical clustering [30] creates a hierarchy of clusters. Algorithms or hierarchical clustering may be *agglomerative*, in which one starts with the individual data points and successively merges clusters together, or *divisive*, in which one starts with one cluster containing all the data points and recursively splits the clusters. Any non-negative-valued function may be used as a measure of similarity between pairs of observations. The choice of which clusters to merge or split is determined by a linkage criterion, which is a function of the pairwise distances between observations. Some common distance measures are:-

- The maxium distance between elements of each cluster (also called complete-linkage clustering)

$$max\{d(x,y) : x \in A, y \in B\}$$

- The minimum distance between elements of each cluster (also called single-linkage clustering)

$$min\{d(x,y) : x \in A, y \in B\}$$

- The mean distance between elements of each cluster (also called average-linkage clustering)

$$\frac{1}{|A|.|B|} \sum_{x \in A} \sum_{y \in B} d(x,y)$$

#### 2.2.1.2 Constrained Clustering

Constrained clustering is a class of semi-supervised algorithm, which typically incorporates either a set of must-link constraint, cannot-link constraints, or both with a clustering algorithm. These constraints encode the relationship between data instances in the dataset. Kiri et. al [36] incorporate these constraints into the popular K-means clustering algorithm.

- **Must-link** constraints specify that two instances have to be in the same cluster
- **Cannot-link** constraints sepcify that two instances must not be placed in the same cluster

In general, constraints may be derived from partially labeled data or from background knowledge abot the domain or the data set. The full set of derived constraints are provided to the clustering algorithm.

### 2.2.2 Mining Tasks in Computer Vision

In this section, we talk about few noticeable works in solving a variety of computer vision tasks using frequent itemset mining.

**Object Detection** The task of object detection involves finding objects which belong to a single semantic class in a sea of images. Data Mining has been employed for the task of mining frequently occurring objects from video by Quack *et al.* [51]. The way in which a transaction is defined is of prime importance when mining frequently occurring objects. A set of central visual words, which were able to survive at least $f_{min}$ frames were obtained, and a transaction was built for each such visual word. The $k$ nearest neighboring visual words, along with their relative location with respect to the central visual word $v_c$ were used. Due to the unexplored nature of the problem, choosing a support threshold is tough. Hence, multiple thresholds are used to run the algorithm, and using a binary split search strategy, the range of the support threshold is fine-tuned until a fixed number of itemsets are not obtained.

Mining consists of, but is not restricted to Frequent Item set Mining. A more general application of mining objects and events from Flickr photos was demonstrated by [52]. Each photograph obtained from Flickr was associated with its geo-spatial coordinates. Photos were clustered, with photographs containing similar objects falling into the same cluster. The Bag-of-Words method was used for image representation, and a spatial verification matching scheme was used to define the similarity between two images. Along with visual cues, text cues obtained from keywords usually associated with photographs on Flickr were further used to facilitate grouping of images.

**Image Classification** Till now, the way in which transactions were created only captured information about the presence or absence of the visual word. Fernando *et al.* [23] proposed a new definition of items, where a histogram of visual words computed over a local image neighborhood is used as an item. The $K$ nearest visual words were used to define the size of the neighborhood, which restricts the total number of visual words in the local histogram Figure2.2. An inherent problem when creating transactions for each feature neighborhood is that a large number of transactions are generated for each image, whereas each image usually contains only a single instance of an object. Hence, the mined frequent itemsets may not always be useful, if they have been obtained from the transactions obtained from only a few set of images. A *Representative Score* was computed for each mined *Frequent Local Histogram*, which ensures that the frequent histograms are evenly distributed across a large number of images in the dataset.

**Feature Selection** Frequent Itemset Mining has been employed for the purpose of identifying features which are discriminative of a particular object class. The utility of mined features has been demon-

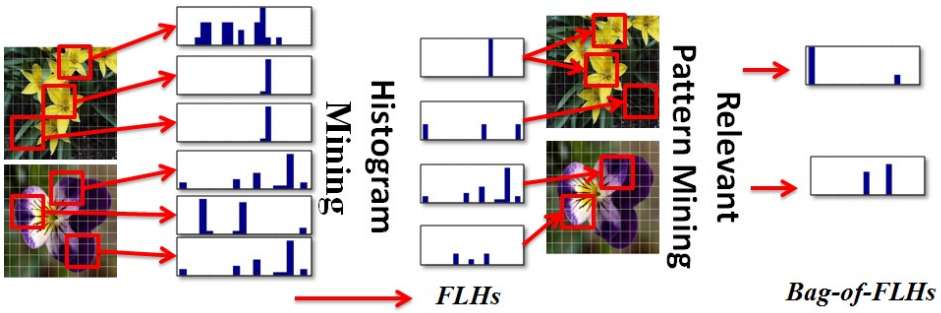



**Figure 2.2** Mining Frequent Local Histograms

strated in classification of objects [49] and actions [27, 26, 25]. Unlike [51], Quack *et al.* [49] created a transaction for a neighborhood around each visual word. Rather than taking the $k$ nearest neighbors, the scale of central feature was used to define the size of the neighborhood, with all visual words lying inside the neighborhood belonging to that transaction. On the other hand, a neighborhood of size $2 \times 2 \times 2$ quadrants across video frames was used to define the transaction in [27]. For the purpose of selecting discriminative features, each transaction is appended with the class label of the object/image from which it was obtained. Following on the same path as the data mining community, association rules were mined which constituted of configurations of features which occurred frequently on instances of the object/action class, and rarely on the background. These association rules were further used for classifying instances of unseen objects in images or actions in video.

### 2.2.3 Challenges in Computer Vision

The problem of mining interesting sets of items, popularly known as Characteristic Pattern Mining, has been explored in various context by the data mining community. A user can be interested in the most frequently occurring itemsets, or items which occur rarely in a dataset. The notion of what makes a pattern interesting is varied. While there are several well-established algorithms for mining interesting itemsets from text data, the same cannot be directly applied to images.

**Items and Transactions:** The notion of what is an item is not very clear when working with images. An item can be a single feature descriptor, or a region in the image, or even a complete image. The definition of an item depends on the problem being solved. Quack *et al.* [51] defines each visual word as an item, and a neighborhood of fixed size around the visual word as a transaction. The itemsets mined from such a dataset are feature configurations. Spatial information can further be incorporated by assigning each visual word a relative location in the neighborhood. Image regions can also be modeled as items, where each region is a semantically meaningful object part. Yao and Fei-Fei [66] model each object (human or musical instrument) as an item, and mine itemsets which represent interaction between humans and musical instruments.

**Uncertainty in Feature Representations:** Items in data mining are discrete and well defined. Counting the occurrence of an item in a dataset amounts to finding all instances of the item identifier in all the transactions in the dataset. However, the same cannot be said for images. Feature descriptors obtained from the same object are susceptible to variation in scale, viewpoint, and illumination. Hence, exact counting of scale-invariant features becomes impossible. The popular method to overcome this problem is to vector quantize the feature descriptors using K-means clustering, and obtain a visual word identifier for each feature descriptor. Similar feature descriptors are assigned the same visual word id, and hence counting becomes meaningful. However, there are several issues associated with vector quantization.

**Interesting Items** Modeling feature neighborhoods typically gives a large number of transactions per image. In such a situation, mining frequent itemsets often gives irrelevant objects such as trees, sky, road etc. which are part of the image background and frequently present in an image. Fernando *et al.* [23] propose a relevance criterion for the mined frequent local histograms, which ensures that the mined histograms are both (a) discriminative, and (b) representative. Thus, histograms which frequently occur over a large number of transactions are retained, whereas those which are frequent only over a small set of images are discarded.

## 2.3 Bag of Words Method

### 2.3.1 Features

The Bag-of-Words method was introduced by Sivic *et al.* [58] for the purpose of retrieving objects from videos. Analogous to the Bag-of-Words (BoW) method for document classification, each image was represented as a document of visual words, each word representing a unique feature. Similarity between images (or documents) was approximated for obtaining a rank list of the images which contain the query object.

The Bag-of-Words method heavily relies on the features which are employed to represent an image. A feature is a unique entity which constitutes an object. For example, we describe a person by his/her facial features, such as the color of the eyes, hair, skin, shape of the nose, face and any noticeable marks. Similarly, objects in an image can be represented by a set of features. Several distinct type of features have been employed in Computer Vision for representing various aspects of an image - Color, Texture, Shape, Geometry, Context and many others. The choice of feature depends largely on the task at hand. For example, classifying images containing sky from those containing grass will work well when images are represented using color features. On the other hand, classifying images containing persons from those containing cars will benefit from shape features.

There are several desirable characteristics of a good feature descriptor representing an image. A feature descriptor should be invariant to the following

- **Position** The position of the object in an image is not fixed. The feature representation of an object in an image should be invariant to its position in the image. This allows us to find all cars in an image dataset, irrespective of their position.

- **Scale** Variation in scale of the object is inevitable. In some images, the object may be captured from a closer distance, while not so much in others. A good feature descriptor should give similar representation for object parts, even at multiple scales.

- **Viewpoint** Similar to scale, the image of an object can also be captured from multiple viewpoints, from one to more cameras. The feature representation of an object should be consistent across viewpoints.

- **Illumination** Image capture is subject to variation in illumination. Some images are captured in bright sunlight, whereas others are captured at night, or indoors, where the amount of light falling on the object is low. A feature descriptor should be able to capture the object parts despite the illumination variation.

SIFT descriptors, which have shown to be invariant to translation, rotation and scale changes [38] is the popular choice of features for many computer vision tasks. These can be computed at multiple interest points generated by one of several algorithms - Hessian-Affine [42], Difference of Gaussian [39], MSER [41]. A dense sampling of the image has shown to give superior performance as compared to the above-mentioned sparse representations, at the cost of increased storage and computation. SIFT Features are computed at regularly sampled points on a grid with a fixed step size, and often at multiple spatial scales.

### 2.3.2 Constructing the Visual Vocabulary

Matching feature descriptors directly across images becomes computationally expensive when working with large datasets, due to (a) large number of descriptors per image, and (b) large number of images. Feature descriptors are usually mapped first to discrete symbols, also called visual words, by means of a clustering technique. Hence, similar descriptors get mapped to the same visual word, completely eliminating the need of a descriptor matching phase.

A visual vocabulary is obtained from a random subset of features sampled from the image dataset. The traditional method of obtaining the vocabulary involves clustering using a method such as K-means. Each cluster denotes a visual word. Hence the visual vocabulary contains a total of K visual words. The size of the visual vocabulary depends on the task at hand. It ranges from a few thousand for object recognition and image classification [15], to one million visual words for instance retrieval [46].

For visual word assignment, the distance from each cluster center is computed, and the visual word identifier of the nearest cluster is assigned to the feature descriptor. The descriptors that are assigned to the same visual word are considered matched.

### 2.3.3 Histogram Representation of Images

We have with us a set of visual words which represent an image. For comparison of images, every image is represented as a histogram of visual words. Each bin in the histogram denotes the occurrence count of the visual word in the image. The number of visual words computed for every image can vary with the size of the image. Hence, it is common practice to normalize the histograms before a similarity is computed. The preferred method of normalization is $L1$ normalization, where each the frequency of each visual word in an image is divided by the total number of visual words present in the image.

## 2.4 Instance Retrieval

The problem of retrieval of the same object from a large corpus of images has received quite some attention in recent years. The goal is, given a query object marked by a bounding box, obtain a rank list of all the images in the dataset, with images containing the object appearing higher in the rank list. The task, however, is not as easy as it appears to be. There are several challenges associated with it

### 2.4.1 Inverted Index

Each image is represented by a vector of frequencies of visual words. Analogous to text retrieval, weighting is applied to the visual word frequencies. The weight of each visual word consists of two terms

- **Inverse Document Frequency:** The inverse document frequency down weights visual words which commonly appear in the dataset, as these words are not very informative. It is computed as

$$idf_i = \frac{N}{n_i} \tag{2.1}$$

  where $n_i$ is the number of images in which the $i^{th}$ visual word occurs in the complete image dataset.

- **Term Frequency:** The term frequency gives higher weight to visual words which appear more frequently in an image, as they describe the image well. It is computed as

$$tf_i = \frac{n_{ij}}{n_j} \tag{2.2}$$

  where $n_{id}$ is the occurrence count of the $i^{th}$ visual word in image $j$, and $n_j$ is the total number of visual words in the same image.

The tf-idf weights are computed as the product of the individual weights. Each image $I$ in the dataset is represented as a vector $V(I)$ of visual word occurrences. The similarity between the query image $I_q$ and a dataset image $I_d$ is defined as the cosine similarity of the two vectors

$$V(I_q).V(I_d)/(||V(I_q)||||V(I_d)|| \tag{2.3}$$

where $||X||$ is the $L_2$ norm of the vector $X$. A rank list of the dataset images is obtained by comparing every dataset image against the query, and sorting the images based on their similarity scores.

### 2.4.2 Spatial Verification

Despite its efficiency in image representation, the Bag of Words method fails to capture the spatial arrangement of visual words in an image, which is essential in representing objects correctly. For instance, while matching images containing a car, the visual words which capture the wheel will always appear in the lower half of the image region containing the car object.

Geometric Verification is used as a post processing step to incorporate spatial information into an otherwise order-less Bag-of-Words (BoW) method. The standard method is to use one of the several variants of the RANSAC algorithm [24] to generate multiple fundamental matrices from randomly sampled subsets of a given set of point correspondences. Based on the number of consistent "inliers", one of the several generated hypotheses is selected. The geometric verification step is computationally expensive, thereby limiting its application to only the top few retrieved results. Several methods have been proposed to incorporate the geometric verification step into the retrieval pipeline itself, rather than being used as a post processing step.

*Chapter 3*

# Prominent Person Mining in Photo Albums

## 3.1 Introduction


**Figure 3.1** Whose Photo Album is this? The answer is obvious to a human being. For a machine, this involved mining of the prominent face from a large number of faces present.

Look at Figure 3.1. Whose Photo Album is this? The answer is obvious – Amitabh Bachchan. If you want a machine to do this task, there are two fundamental problems to be solved. (i) Find the person (face image) who owns this album from a large number of faces present. (ii) Label the person with a name based on apriori knowledge. In this work, our aim is limited to finding the person (face image) to whom a photo album belongs. We focus our attention on finding the template or representative of the person in the form of a face image.

In photo sharing sites, we often find the owner posing for pictures with other people, and posting them as an album. In most cases, he/she occurs the most number of times in these pictures, and is the most prominent person. Hence we call our technique as that of prominent person mining. This could be a step towards extracting useful information from unannotated large collection of images and videos

available on the Internet. Such methods will also have applications in processing surveillance videos for mining social and behavioural patterns of people the camera frequently observes.

The notion of prominent person mining is different from that of Frequent Itemset Mining (FIM) [4]. In case of FIM, the itemsets are discrete and well defined. However, in our case the "items" or "persons" appear with wide variations. This uncertainty has to be accomodated into the mining algorithm. In addition, the prominent person is not just the frequent person. Prominent person often occupies a significant area of the image. He/She is also the centre of attraction of the image.

There have been earlier attempts at using data mining techniques in computer vision. Sivic and Zisserman used video mining in [60] for obtaining the principal objects, characters and scenes by measuring the re-occurence of spatial configurations of viewpoint invariant features. Frequently occuring spatial configurations were found using clustering algorithms, rather than any frequent itemset mining schemes. Quack et al. find frequently occuring objects and scenes in [51], where frequent itemset mining schemes have been used. Data mining algorithms have also been used to assist tasks like object detection [49] and mining objects and events from community photo collections [52]. The characteristic persons are mined in a movie in [33] by first mining the characteristic scenes, and then the persons present in those scenes.

We differ from previous works in multiple dimensions.(i) We focus our attention on prominent person mining rather than frequent itemset mining,(ii) We refine the existing data mining algorithms to suit the problem of prominent person mining, and (iii) Our mining algorithm works under the uncertainty of feature representation rather than vector quantizing the feature descriptor to obtain a crisp representation of visual entitites.

We formulate our problem as that of finding the most prominent person in the given set of images. We use the eigenface representation as described in Section 2. In Section 3 we mine prominent persons using an Apriori-like algorithm which generates prominent candidiates in $k$ dimensions from prominent persons in $k-1$ dimensions in the eigenface space. We conduct extensive experiments on multiple datasets with images captured in real world scenarios and achieve excellent results. We test our approach on 150 photo albums consisting of over 2400 images download from the Internet and achieve an accuracy of over $80\%$

## 3.2 Face detection and representation

Faces are extracted from the images in the album using the OpenCV implementation of the Viola-Jones face detector [64]. For the process of mining, each face image is represented in the form of a feature vector, in which each value corresponds to the contribution of one Eigenface. Given below are the steps for representation [62].

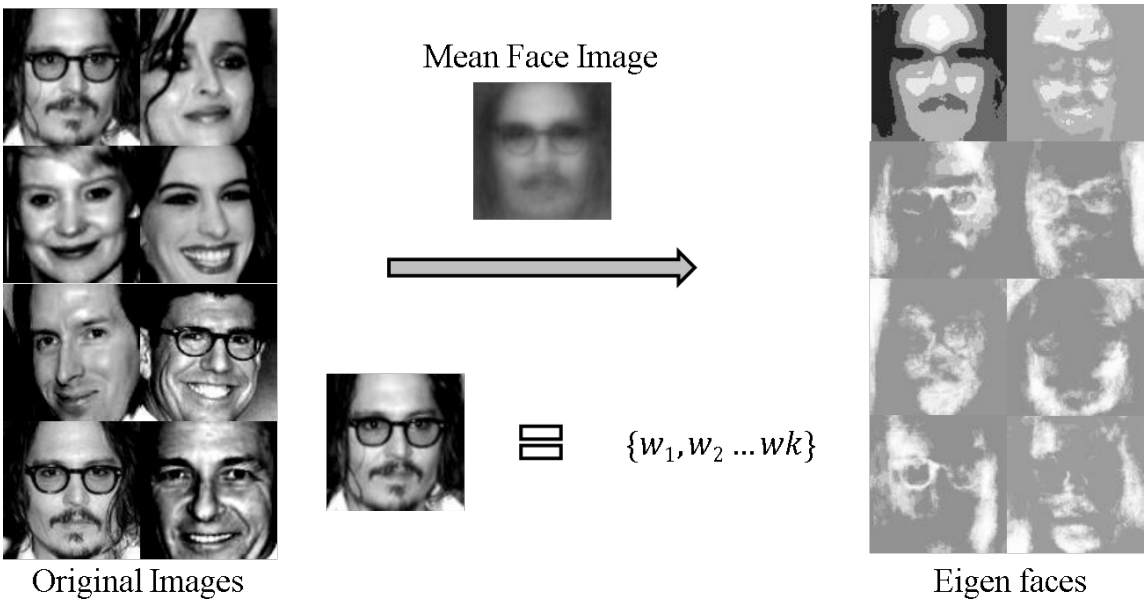



**Figure 3.2** Obtaining Eigen Faces from set of Original Face Images.

### 3.2.1 Eigenface

In the first step, each of the *M* images, $\Gamma_{1-n}$ is transformed into a vector of size N. The mean image $\Psi$ is calculated as

$$\Psi = \frac{1}{M}\sum_{n=1}^{M}\Gamma_n$$

We then compute the difference $\Phi_i$ between each image and the mean image as

$$\Phi_i = \Gamma_i - \Psi$$

Next we seek a set of *M* orthogonal vectors, $u_{k,k=1-M}$, which best describes the distribution of the data. The $kth$ vector $u_k$, is chosen such that

$$\lambda_k = \frac{1}{M}\sum_{n=1}^{M}\left(u_k^T\Phi_n\right)^2$$

is maximum, subject to

$$u_l^T u_k = \delta_{lk} = \begin{cases} 1 & \text{if } l = k \\ 0 & \text{otherwise} \end{cases}$$

Here, $u_k$ and $\lambda_k$ are the $k^{th}$ eigenvectors and eigenvalues of the covariance matrix C, which is computed as

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = AA^T$$

where $A = [\Phi_1, \Phi_2 \ldots \Phi_M]$

We now want to compute the eigen vectors $u_k$ of $AA^T$, but since the matrix $AA^T$ is very large, the eigenvectors $v_k$ of $A^TA$ are computed and then used to obtain $u_k$ using the relation $u_k = Av_k$. We choose the eigenvectors corresponding to the top few eigen values as eigenfaces. We now project the mean-subtracted image on the eigenfaces and store the result in a weight vector $\Omega = [w_1 \ldots w_M]^T$, where $w_k = u_k^T (\Gamma - \Psi)$ is the weight value corresponding to the $k^{th}$ eigenvector. Each such weight vector is a transaction in our transaction database which is used for prominent person mining.

## 3.3 Prominent Person Mining

Our goal is to find the most prominent person from the face images of persons provided to us by the face detector. Our database consists of several transactions, each being a face in the set of extracted faces from the album images. Each face is a point in the *K* dimensional eigenface space. We propose a greedy algorithm to find the most prominent person in a subset of dimensions. We first mine the most prominent person in a single dimension and move in a heirarchical manner to mine prominent persons in more than one dimensions. Our approach is based on a popular data mining problem of mining frequent itemsets. Here we briefly summarize the terminology and methodology of frequent itemset mining and other techniques which have been used in our algorithm.

### 3.3.1 Approach

In traditional frequent itemset mining, the itemsets are well defined and discrete. To find how frequently an apple is bought in a given set of transactions, we count it's occurence in the transaction database. This cannot be done with faces, as "face" representations can vary. We use an approach similar to the Apriori algorithm, where we change the notion of frequent itemsets, and refer to them as prominent persons, taking into account the uncertainty in faces. Prominent "persons" or "itemsets" are faces of the same person who is prominent in the given set of images. Prominent persons often occupy a significant area of the image. He/She is also the centre of attraction of the image. We first find the most prominent person in a single dimension, and then move towards more number of dimensons in the eigenface space. Here each dimension corresponds to one eigenface used for computing the weight values. Our algorithm takes three input parameters:-

- $K$ (Number of dimensions): Minimum number of dimensions across which we want the person to be prominent.
- $D$ (Diameter): Diameter for the terminating condition of the agglomerative hierarchical clustering method used to find prominent persons in a single dimension.
- $S$ (Minimum Count): Minimum support threshold used in frequent itemset mining schemes to determine whether an itemset is frequent or not.

### 3.3.2 Finding prominent persons in 1 dimension

In this step, we wish to identify prominent persons in each dimension. We perform semi-supervised agglomerative hierarchical clustering in each dimension. This method builds the hierarchy from individual elements by progressively building clusters. In every step the clusters which are closest to each other are merged until some terminating criteria is met. The distance between two clusters can be defined in several ways. We perform complete linkage clustering, where the maximum distance between elements of each clusters is used. It is defined as

$$max\{d(x, y) : x \in A, y \in B\}$$

where $A$ and $B$ are the clusters to be merged. All clusters with diameter less than the diameter threshold $D$ are computed. Semi-supervision comes in the form of cannot-link constraints. We make sure that faces which have been obtained from the same image cannot belong to the same cluster. This constraint is justified as the faces which have been extracted from one image are bound to be of different person, and only one of them can be the most prominent person. The elements belonging to the cluster with the largest number of items are considered as prominent in this particular dimension if the number of elements in this particular cluster is greater than the provided minimum count threshold $S$. We assume that the person to which this profile belongs is present in every image in his album. Hence if we have N test images, and of those $M(\gg N)$ face images are extracted, our minimum count theshold is set to a value $\frac{N}{3}$ to take care of missed detections by the face detector. In an ideal scenario, where the face detector gives $100\%$ results, and the environment in which the face images have been captures is controlled, the value of $S$ can be set closer to $N$.

### 3.3.3 Finding prominent persons in *k* dimensions

Each dimension of the eigenface space is represented by a bit vector, where the bits corresponding to the prominent persons are set to 1. Say, if we have 10 transactions in our database, and the bit vector for the $i^{th}$ dimension $d_i$ is $\vec{d_i} = 1100010000$, then the $1^{st}$, $2^{nd}$ and $6^{th}$ persons are prominent in dimension $d_i$. We progressively compute prominent persons in $k$ dimensions. This is done in an Apriori fashion, where candidiates for prominent persons are generated in $k$ dimensions from prominent persons in $k-1$ dimensions. Hence, if a person is prominent in dimensions $d_1d_2$, $d_1d_3$, $d_2d_3$ and $d_2d_4$, candidate triplet of dimensions which are generated are $d_1d_2d_3$ and $d_2d_3d_4$. The bit vectors are obtained by taking and of the bit vector of each individual dimension.

$$d_1d_2 \ldots d_i = \vec{d_1} \ \wedge \ \vec{d_2} \ \wedge \ \ldots \vec{d_i}$$

In the pruning step, we eliminate candidate dimensions in which the person is not prominent in at least one of the immediate subset of dimensions. For example, we eliminate $d_2d_3d_4$ from the candidate set since it's immediate subset $d_3d_4$ does not contain a prominent person. Our algorithm returns persons who occur at least S times in atleast K dimensions.

### 3.3.4 Semi-supervised Prominent Person Mining

Till now, our notion of prominence was based on the occurence frequency of a person in the given image set. However, prominent persons often occupy a significant area of the image and are usually the centre of attraction. We modify our definition of prominence as follows:-

- The most prominent person occurs the maximum number of times in the photo album.
- The most prominent person, to whom the photo album belongs, is often the centre of attraction in the images in the photo album. For every face detected, we find out the distance of the centre of the bounding box for that face from the centre of the image along the horizontal direction.
- The most prominent person often occupies a large area in the images in the photo album. For every face detected, we store the ratio of the size of the bounding box for the face to the total size of the image.
- The lesser the number of people in a given image, the higher is the probability of each being a candidate for the most prominent person. In all photo albums, the are some photos which have only the owner of the photo album in it. Hence, for each face we store the number of persons in the original image from which the face was extracted.

All this information is stored while running the OpenCV impementation of the Viola Jones face detector to detect and segment out faces. We propose a semi-supervised version of prominent person mining which incorporates the above definition of prominence.

#### 3.3.4.1 Incorporating semi-supervision into the mining process

We successfully incorporate semi-supervision into the mining process. Each face extracted from an image is assigned a weight based on the above-mentioned definition of prominence:-

$$w(f_i) = \begin{cases} f_i.presence + f_i.ratio - f_i.deviation & \text{if } w(f_i) > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $w(f_i)$ is the weight assigned to face $f_i$. $f_i.presence$ encodes information about the number of persons present in the image and is calculated as

$$f_i.presence = \frac{1}{N}$$

where N is the number of persons in the image in which face $f_i$ was detected. Hence, if a face detected in an image is the only face in that image, it'll be given higher priority as compared to one of multiple faces detected in another image. $f_i.ratio$ encodes information about the area occupied by the boudning box of a detected face in the image. It is computed as

$$f_i.ratio = \frac{BB(f_i).width * BB(f_i).height}{Size\ of\ Image}$$

where $BB(f_i)$ is the bounding box of face $f_i$. $f_i.deviation$ encodes information about how far the detected face is from the centre. It is computed as

$$f_i.deviation = \frac{|0.5 * ImageWidth - BB(f_i).centreX|}{0.5 * ImageWidth}$$

Note that while the first two terms have a positive sign, the third term has a negative sign associated with it in the weight computation. This is because we want the first two terms used in the weight computation ($f_i.presence$ and $f_i.ratio$) to be maximum. The most prominent person incurs a penalty for being at a greater distance from the centre. Hence we want this distance, the third term in the weight computation ($f_i.deviation$) to be minimum. The count of a candidate prominent person in a single dimension or combination of dimensions is given by

$$Count(p_i) = \sum_{i=1}^{I} w(f_i)$$

Each component of the weight term is normalized to ensure equal contribution in the mining process. This is done using the *min-max normalization* technique, which performs a linear transformation on the original data. Suppose $min_{f_i}$ and $max_{f_i}$ are the minimum and maximum values of one component of a weight value. It maps the component value $f_i.x$ in the range [0,1] by computing

$$f_i.x' = \frac{f_i.x - min_{f_i}}{max_{f_i} - min_{f_i}}$$

We pause here to understand what happens when such weights are associated with transactions in a database during frequent itemset mining. As an example, consider a database which consists of transactions occuring in a departmental store. There are two types of transactions, one which occur duing the period when there is a sale at the store. The second type of transactions are those in which fresh stock is sold. While analysing such a database for products which are more frequently sold, the store owner will be more interested in finding out products out of fresh stock which are frequently sold, rather than those which are sold during the sale. This is because the former will generate a higher profit for the store. Hence, the transactions in which fresh stock is sold is much more important and can be assigned a higher weight. Similarly, we are more interested in faces, which intuitively belong to the most prominent person. Hence, we try to derive a weight which will be larger for the faces of the prominent person we're trying to mine.

Our approach can be compared to Weighted Association Rule Mining in [22], where three types of weights have been introduced into the mining process. These are

- **Item Weights:** *Item weight* is a value attached to an item representing its significance.
- **Itemset Weights:** Based on the item weight, the weight of an itemset can be derived from the weights of its enclosing items.
- **Transaction Weights:** It is a value attached to each of the transactions. Usually the higher the transaction weight, the more it contributes to the mining result.

We use Transaction weights in our algorithm, where each transaction represents a face, and the weights are computed from the intuitive definition of prominence.

We might get multiple results if the minimum count threshold, $S$ is low. This usually happens in cases where each image has a large number of people. We assign a primary score,

$$PScore(p_i) = k_i$$

to each prominent person returned by our algorithm, where $k_i$ is the number of dimensions in which person $p_i$ is prominent with frequency $s_i$. If the primary score for the results returned are the same, we assign a secondary score,

$$SScore(p_i)_k = s_i$$

to each prominent person in the same dimension, where person $p_i$ is prominent with frequency $s_i$ in k dimensions. Hence we prefer those results in which the person is prominent in the dimensions whose eigenvalues are larger. For the prominent persons across the same number of dimensions, we return the person with the maximum frequency. We compute a score similar to the weights assigned to each face in the prominent person mining method. The score of each face is computed as

$$TScore(f_i) = f_i.presence + f_i.ratio - f_i.deviation$$

For the result with the maximum primary score ($PScore(p_i)$ and secondary score ($SScore(p_i)$), we return the face with the maximum value of $TScore$.

## 3.4 Experiments and Results

**Table 3.1** Table summarizing various data sets used for our experiments on mining the most prominent person in a given photo album.

| | ORL | LFW | IMDB | Facebook |
|---|---|---|---|---|
| # Images | 400 | 1500 | 1780 | 947 |
| # Subjects | 40 | 143 | 25 | 26 |
| # Photo Albums | 100 | 91 | 100 | 40 |
| | Dark homogeneous background, Upright frontal position | Non-uniform background, Variation in face pose | High quality images, Large groups of people | Low quality images, Large groups of people |

We conducted extensive experimentation on several datasets to demonstrate the effectiveness of our approach in real world scenarios. Table 3.2 summarizes the various datasets that we have used for evaluating the performance of our algorithm in different scenarios. The first set of experiments were

conducted on face images taken in a controlled environment, with small variation in appearance and background. For this purpose, we used the ORL face database. The ORL face database consists of ten images each of 40 different subjects. The size of each image is $92 \times 112$ pixels. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open/closed eyes, smiling/not smiling) and facial details (glasses/ no glasses). All images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement). We generated 100 photo albums from this set, with the most prominent person having 8-10 photos and the other persons with 1-4 photos. In the eigenface computation, the eigenvectors corresponding to the top 10 eigenvalues were used for the face representation. Using more number of eigenvectors does not affect the results as most of the the information about the distribution in the data is stored in the first few eigenvectors. Out of the 100 photo albums we tested our approach on, we found 100 prominent persons. For 90 of these photo albums the person we found as most prominent was the actual owner of the photo album. The results of these experiments are summarized in Table 3.3 Column II.

Now that we have estabilished the correctness of our approach on images captured under controlled conditions, we needed to test on a real world scenario. Characteristic of real world images are change in appearance of a person, variation in pose and background. For this we downloaded images of the Labeled Faces in the Wild (LFW) dataset. The LFW Database [1] consists of more than 13000 images collected from the web. 1680 of the people pictured have two or more distinct photos in the data set and only 143 have more than 10 images. We downloaded this subset of images of 143 people. The database has been designed for the task of unconstrained face recognition and is much more challenging than the ORL face dataset. Most of the images of the same person have been taken from different event gatherings, often separated by a large time gap, leading to change in apperance, pose and illumination. The OpenCV implementation of the Viola-Jones face detector was used to segment out the faces in each image, which were resized to a fixed size of $100 \times 100$. We then generated 92 photo albums in a similar fashion as in the case of ORL face dataset and conducted experiments to find out the most prominent person in each album. The results of the experiments are summarized in Table 3.3 Column III.

**Table 3.2** Table summarizing statistics of different datasets used.

| Dataset | No. of Images | No. of Photo Albums | Avg Photos per Album | Avg Faces per Album |
|---|---|---|---|---|
| ORL | | 100 | | 18.71 |
| LFW | | 92 | | 18.6 |
| IMDB1 | 1780 | 100 | 17.8 | 32 |
| IMDB2 | 2575 | 50 | 51.51 | 73 |
| Facebook | 943 | 40 | 23.57 | 103.1 |

Our next task was to test our algorithm on real world photo albums. For this purpose, we crawled pages of 25 celebrities on IMDB and downloaded 100 photo albums. Each celebrity page has a link to multiple pages for each event he/she has been a part of. On that page are photos taken during that particular movie premiere or award show. On an average, each album contains 16 photos. The faces

**Table 3.3** Results without semi-supervision

| | ORL | LFW | IMDB1 | IMDB2 |
|---|---|---|---|---|
| Number of Photo Albums | 100 | 92 | 100 | 50 |
| Number of Prominent Persons found | 100 | 92 | 100 | 50 |
| Correct Prominent Persons found | 90 | 70 | 75 | 42 |
| Accuracy (in $\%$) | 90 | 76 | 75 | 84 |

were extracted using the OpenCV implementation of the Viola-Jones Face Detector. 29 faces were detected per photo album on an average. The faces were resized to a fixed size of $100 \times 100$. The faces in each photo album were used for the computation of eigenfaces for that album. The images were represented using eigenfaces corresponding to the top 10 eigenvalues. Out of the 100 photo albums, the most prominent person mined in 75 albums was the actual owner of the album as shown in Table 3.3 Column IV. We use the term IMDB1 to denote this set. The value of $N$ is lesser than that used for the ORL dataset to account for false positives and false negatives by the Face detector.

The decrease in accuracy from the ORL face dataset to the Labeled Faces in the Wild dataset and albums downloaded from IMDB can be attributed to the variation in background and pose of the actor. In few of the cases we detected the second most prominent person in the photo album, which was usually when the number of face images in the album were small and the number of images of the most prominent person comparable to that of the second most prominent person.

We vary various parameters to see how they affect our result. Figure 3.5 shows the variation of the number of prominent persons found with the increase in the value of K, that is the number of dimensions across which results are returned for the IMDB 1 Test set. We can see that as we tighten the constraint on the number of dimensions, the number of prominent persons found decreases, with a value of only 9 for $K = 7$. The accuracy also decreases in a similar fashion as illustrated in Figure 3.6. However, out of the total prominent persons found, the percentage of correct prominent persons found increases with increase in the value of K as shown in Figure 3.7. This shows that the greater the number of dimesions across which results are returned, the higher is the chance of them being accurate.

We also perform experiments to find out how each of the three factors, that is

- Location of the person in the image,
- Area occupied by the person's face in the image, and
- Number of persons present in the image

which are used in introducing semi-supervision into the algorithm, indivudally affect the performance of our algorithm. This enables us to get an idea as to which of these factors are more important in correctly finding out the most prominent person.

In the Eigenface representation, eigenvectors corresponding to top few values capture most of the distribution in the data. We observed that this was the case and the dimensions in which a person was prominent were from 1-5. Hence, using dimensions 5-10 were proving to be redundant in most cases

as they were not contributing to the final result. To reinforce this theory and ensure that all the 10 dimensions played a role in finding the most prominent person, we crawled pages of 200 celebrities and downloaded photo albums. Of these, we kept those which had more than 50 face images after running the Viola Jones Face detector and conducted experiments. On 42 out of the 50 albums we tested on, we successfully mined the correct prominent person. The results on this dataset (IMDB2) have been shown in Table 3.3 Column V, and are in accordance with our theory.

We also evaluate the performance of the semi-supervised version of our algorithm on the two datasets downloaded from IMDB. The algorithm shows improved performance as compared to the unsupervised algorithm. The accuracies jump to over $90\%$ for both these datasets. We further investigate on how each of the three factors (Location of face/Area of face/Number of persons in Image) which have been considered individually affect the performance of our algorithm. This will help us to understand which of the above-mentioned factors plays an important role as compared to the others in mining the most prominent person. Table 3.4 summarizes the contribution of each factor to the final result. From it, we can infer that even though each factor does not individually contribute significantly in greatly enhancing the performance of the prominent person mining technique, using a combination of all three helps us to achieve a high accuracy. This can be attributed to the fact that not all these factors might hold true for the most prominent person in most of the images. In photographs of a group of people, the most prominent person can be found at different positions rather than centre. Also, in some cases where the ratio of the size of the bounding box around the prominent person's face to that of the size of the image is high, there is a high chance that same will be so for other people present in the same image. Hence, using a combination of all these factors proves to be useful for us.

**Table 3.4** How each factor affects results on IMDB1 Test Set. P = Presence (1/Number of Persons in Image), A = Ratio of Area of Face Bounding box to Area of Image, L = Ratio of Distance from Centre to Width of Image

| | PAL | P | A |
|---|---|---|---|
| Number of Prominent Persons found | 100 | 100 | 100 |
| Correct Prominent Persons found | 91 | 82 | 81 |
| Accuracy (in $\%$) | 91 | 82 | 81 |

**Table 3.5** Results with Semi-supervisionn

| | IMDB1 | IMDB2 | Facebook |
|---|---|---|---|
| Number of Photo Albums | 100 | 50 | 40 |
| Number of Prominent Persons found | 100 | 50 | 40 |
| Correct Prominent Persons found | 91 | 47 | 24 |
| A (in $\%$) | 91 | 94 | 60 |

Lastly we experment on 40 photo albums downloaded from Facebook. These albums pose a great amount of challenge as compared to the other photo albums which we tested on due to the following factors:-

- **Low quality images:** Images captured and put up on social networking sites are usually of low quality, taken from a standard digital camera. Seldom does the person who captures the photograph take care of factors such as lighting and blur, which greatly affects image quality.
- **Large number of people:** Each of theses images also possess a large number of people, making the problem of prominent person mining much harder. This also results in smaller size of the face as compared to the size of the image, which results in difficuly in processing for mining tasks.

Figure 3.3.4.1 demonstrates visual examples of challenges which are inherent when mining from photo albums downloaded from Facebook. We mined the most prominent person using the semi-supervised version of our algorithm. Out of the 40 photo albums, we found the correct prominent persons in 24. On carefully evaluating the results, we observed that our algorithm didn't perform as expected when the photo album contained images of multiple events taking place over a long period of time, where there is high amount of variation in the appearance of person (hat/no hat, glasss/no glasses, hair style) and pose. On the other hand, the results are good on the subset of these albums which have images from a single event/outing, albeit under varying illumination conditions and number of people posing for the photograph. Table 3.5 summarizes the performance of semi-supervised prominent person mining on the IMDB1, IMDB2 and Facebook datasets.

## 3.5 Summary

In this chapter, we have introduced a new method of finding out the owner of a photo album, who is often the most prominent person in the set of album images. For this, we have extended the frequent itemset mining technique to mine prominent persons based on our new definition of prominence for a person. We have also developed a semi-supervised version of the algorithm, where we take into account several factors which define prominence. The algorithm has been thoroughly evaluated on different challenging scenarios to ensure mining of the correct owner of the photo album.

rint


**Figure 3.3** Prominent Persons mined on photo albms of six celebrities downloaded from IMDB. A few images frome each photo album are shown on the left. The corresponding most prominent person mined is shown on the right (a) Actor: Nicholas Cage, Movie: Matchstick Men (b) Actor: Jennifer Anniston, Movie: Marley and Me (c) Actor: George Clooney, Movie: Ocean's Eleven (d) Actor: Leonardo Di Caprio, Movie: Blood Diamond (e) Actor: Tom Cruise, Movie: Mission Impossible III (f) Actor: Jennifer Connely, Movie: Reservation Road

(a) People often upload photographs of outings with large groups of people. In such photo albums, the face of the most prominent person is very small and is present along with a large number of other people's faces. Mining the most prominent person becomes tough in such a situation

(b) Another example of photo album where large group of people is present. This is much more challenging than the previous photo album

(c) Not all photographs uploaded in a photo album will contain the complete view of the face of the most prominent person. For example, in the above album downloaded from Facebook, the most prominent person is only partly visible in the 4th and 5th photographs, making him difficult to mine. The fact that this goes against our criteria of the most prominent person appearing in the center of photographs also makes it much harder

(d) In the above photo album, while the most prominent person is clearly visible, and also appears in the centre of photographs, the faces are occluded by goggles worn by all the people in the pictures. This makes it harder to match faces across images, as the eyes form an important and large part of the facial region.

(e) There can be large variation in facial expression as well as pose of the face of the most prominent person, due to which matching faces across images becomes tough, leading to difficulty in mining.

(f) Often the person capturing the photographs which are uploaded on a social networking website does not pay attention to detail such as focus, lighting etc. This leads to poor quality photographs (blur, dim or bright light). Mining in such a setting is tough

**Figure 3.4** Various Photo Albums constituting our dataset of photo albums obtained from Facebook. We demonstrate why mining from such albums is hard.

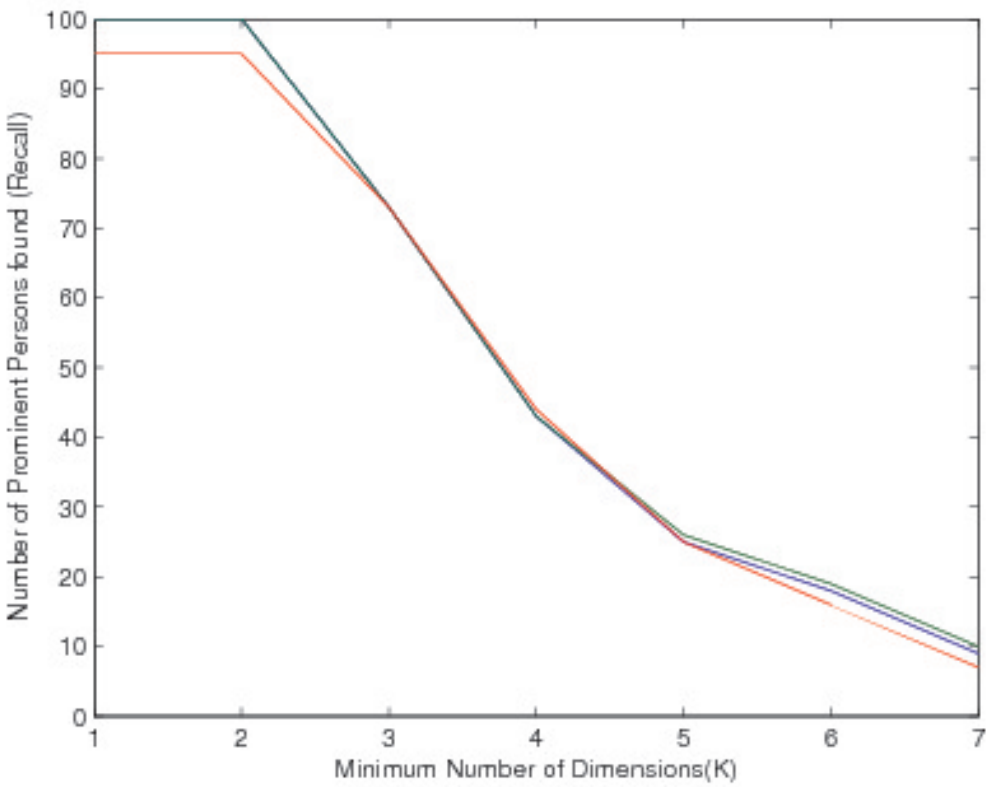



**Figure 3.5** Plot showing variation in Number of prominent persons found (Recall, Y-axis) with increase in value of Minimum number of dimensions (K, X-axis) across which results are returned on the IMDB1 test set. The Recall decreases as we tighten the constraint on K. The Black, Blue and Red curves denote the recall curve in cases when 10, 15 and 20 eigenvectors are used for computation of eigenfaces.

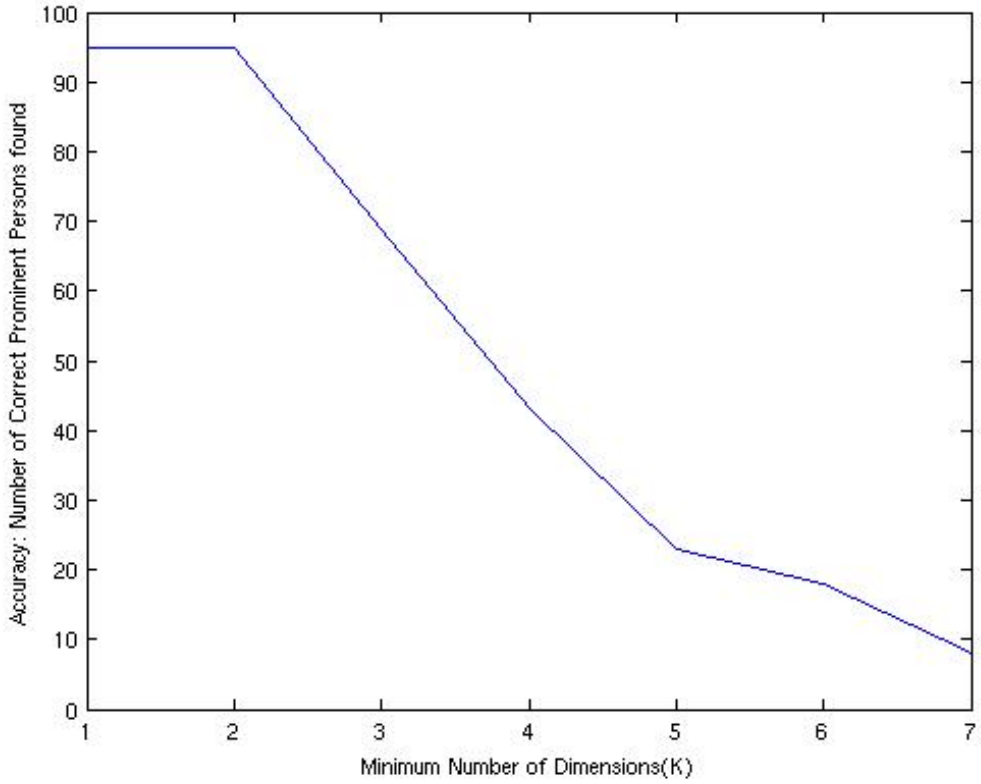



**Figure 3.6** Plot showing variation in Number of correct prominent persons found (Accuracy, Y-axis) with increase in value of Minimum number of dimensions (K, X-axis) across which results are returned on the IMDB1 test set.

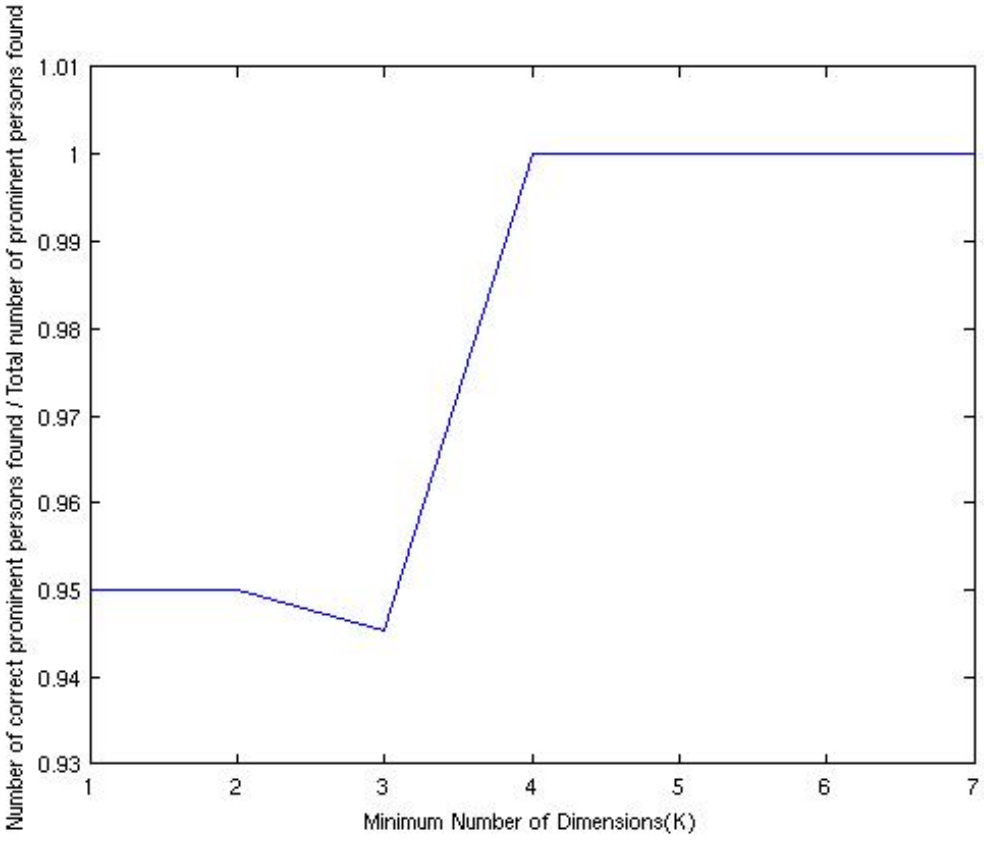



**Figure 3.7** Plot showing variation in Percentage of correct prominent persons found from the total prominent persons found (Y-axis) with increase in value of Minimum number of dimensions (K, X-axis) across which results are returned on the IMDB1 test set. As the constraint on the number of dimensions are tightened, the percentage of correct results returned increases.

*Chapter 4*

# Mining Characteristic Features for Architectural Style Categories

## 4.1 Introduction

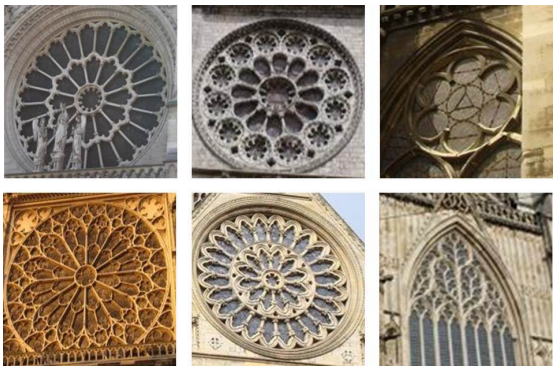


**Figure 4.1** Monuments of an Architectural style have visually similar structures in common. These are some commonly occurring structures mined from our dataset of European Architectural Styles for Gothic Architecture.

In recent years, the dedicated efforts of a considerable section of the computer vision community have been directed towards solving the problem of instance retrieval. Given a query image or region, the goal is to find and retrieve identical instances of the query object from a large collection of image/videos. The work on Oxford5k dataset[46] has contributed significantly in this direction. Google Goggles is a widely popular product of research in instance retrieval. Sivic *et. al* [59] introduced the Bag-of-Visual-Words (BoW) method to search for objects and scenes in feature films. Images are represented using a histogram of visual words, obtained by clustering high dimensional descriptors obtained from images. Similar images are retrieved using techniques such as inverted file index [59] or min-hash based methods. A key addition to this pipeline, was the introduction of spatial verification for incorporating geometric information into the orderless Bag of Words method. Initially used as a post-processing step [59, 46], it is now an integral part of the retrieval pipeline [70, 34] for matching multiple views of the same object across images. Since then, the BoW technique, in its many forms, has become a mainframe

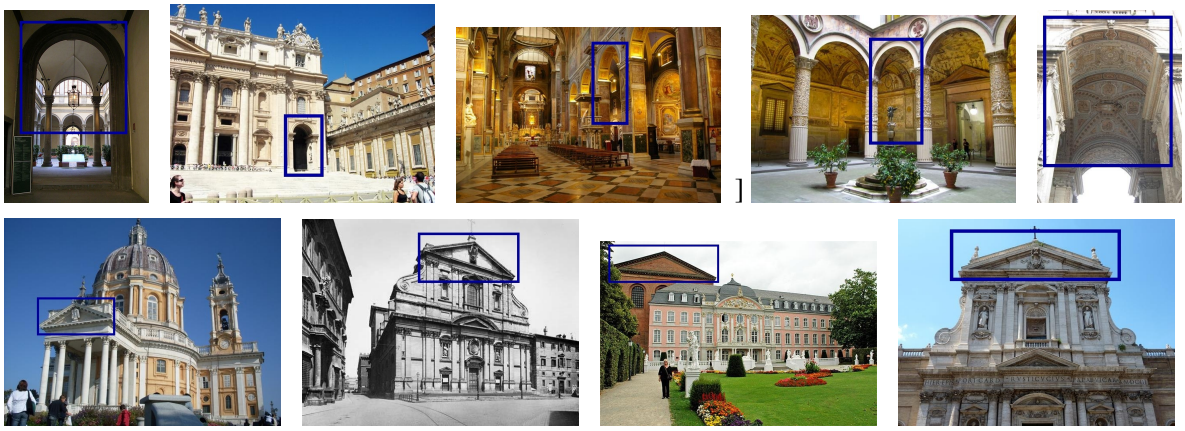


**Figure 4.2** Characteristic Features for two categories of Architectural Styles. Row 1: (Left to Right) Palazzo Strozzi (Florence), San Pietro, Sant Agostino, Santa Maria (Novella), and St. Peter's Basilica. Semi-circular arches are common features of monuments of Renaissance architecture. Row 2: (Left to Right) Basilica di Superga, Church of the Gesu, Elector's Palace, and Santa Susanne (Rome). Monuments in Baroque Architecture often have a triangular pediment above the main entrance. These and many other structures are common across different buildings, belonging to similar architectural styles. Unlike instances, these structures are found in different buildings and are characteristic features for that class of architectural style.

of several instance retrieval techniques. For the purpose of smooth object retrieval, Arandjelović and Zisserman [5] proposed a Bag-of-Boundaries (BoB) method by vector quantizing HOG descriptors computed on regularly sampled points from object contours. Moving on from objects, several works [46, 48, 70] have used the Bag of Words method for searching on building facades and architectural features. In all these cases, the authors find instances of the same building, albeit, from multiple viewpoints and with visual ambiguities. We ascend one level higher, and pose the question - Are buildings only instances?

According to Wikipedia, an architectural style is a specific method of construction. This may include elements such as form, materials, arrangement, and regional characters. Just like fashion trends, architectural styles vary with time, as well as geographical region. We believe that buildings, apart from being instances, can also be categorized by architectural styles. Each architectural style has certain features that make it distinguishable from other forms of architecture. These characteristic features play a key role in the identification of such monuments. For instance, Rose windows[1] are often found on monuments of the Gothic architectural style. Figure 4.1 shows images of different varieties of Rose Windows mined as part of our results from various Gothic monuments. An expert in this field provided with information about the characteristics of a monument can easily identify the category of architectural style to which this monument belongs, if not the monument itself. Given a large collection of images of various monuments, we are interested in automating the task of identifying such features and finding an answer to the possibility of buildings being something more than instances. We have no

[1] http://en.wikipedia.org/wiki/Rose_window

apriori information about (a) which monuments have been captured in the images that we possess, and (b) what features are we trying to discover.

We start by exploring the utility of configurations of low-level discriminative features in categorizing buildings into separate architectural style categories. Each style of architecture is characterized by several features. The color and texture of buildings comes from the materials used for construction. Structures such as windows and arches, vaults and domes can vary in shape across monuments. Various engravings and decorations found on monuments are often characteristic to the time period when they were built, and may reflect the lifestyle and ideas of the people who were around when the monument was being constructed. Based on these observations, we experiment with multiple features to capture color, texture, shape and appearance information, and create strong baselines.

Being able to automatically identify visually characteristic elements in architectural scenes can open up the possibility of a whole new area of research. Such characteristics can be used to assist tasks in classification and retrieval of architectural style categories. As a stepping stone towards solving this problem, we propose a simple, yet effective approach to mine characteristic pairs of features which occur frequently across buildings in the same category and use them for improving classification performance. In all our experiments, we ensure that images of the same monument cannot be used for training as well as testing. This ensures that the results we obtain are not instances, but different buildings with similar architectural styles. The improvement in categorization further motivates us to explore higher-level, semantically meaningful features.

Research on characteristic features for architectural scenes is a fresh research direction. A new type of local descriptor, computed at symmetric points in images, has been proposed for matching architectural scenes in [31]. Doersch et al. [19] find visually repeating elements (windows, balconies and street signs) from Geo-tagged imagery of the city of Paris obtained from the Internet. Similar characteristics are obtained for different cities and those which are specific to Paris are identified. Using a patch-based method for finding visually-informative elements as in [19] restricts the features that are discoverable. The problem of discovering characteristic features across monuments is much more challenging due to (a) Multiple scales of characteristic features (b) Significant variation in appearance of the same visual characteristic across monuments (c) Multiple viewpoints and occlusion. Unlike [19], where discriminative patches containing small elements such as windows and street signs were mined, we mine characteristic features at multiple scales which are semantically meaningful and informative of a particular architectural style category. These characteristics can range from capitals, found in Renaissance Architecture[2] to huge windows with pointed arches found on Gothic monuments[3]. Characteristic features, represented by frequent sub-graphs, were mined using in [16], and used for improving classification of architectural scenes and product images. We categorize images into one of several categories of architectural styles. Our dataset comprises of large number of high resolution images of various mon-

[2]http://en.wikipedia.org/wiki/Renaissance_architecture
[3]http://en.wikipedia.org/wiki/Gothic_architecture

uments obtained from the Internet. The extent of characteristic features of architectural styles, if they do exist, is unknown.

## 4.2 Image Categorization


**Figure 4.3** Monuments of various Architectural Styles from our database of European Monuments. Row 1 - Art Noveau, Row 2 - Baroque Architecture, Row 3 - Gothic Architecture, Row 4 - Renaissance Architecture, 5 - Romanesque Architecture

Being able to correctly discover characteristics which distinguish an architectural style category has several applications. These features can be used to assist classification and retrieval tasks in Computer Vision. We perform a set of experiments to show how characteristic features can be used to improve

**Table 4.1** European Monuments Dataset

| **Art Noveau** | **Baroque Architecture** | **Gothic Architecture** |
|---|---|---|
| Casa Batllo | Basilica de Superga | Abbey of St. Denis |
| Casa Lleo Morera | Church of the Gesu | Chartres Cathedral |
| Casa Mila | Elector's Palace | Notre dame de Paris |
| Elisabeth's Church | Queluz National Palace | Salisbury Cathedral |
| Sagrada Familia | Santa Susanne,Rome | York Minster |
| **Renaissance** | | **Romanesque** |
| Florence Cathedral | | Mainz Cathedral |
| San Pietro | | Mosiac Abbey |
| Sant Agostino | | Notre dame de Puy |
| Santa Maria, Novella | | Peterborough Cathedral |
| St. Peter's Basilica | | Sant Ambrogio |

classification of an architectural style category. This can be a stepping stone towards using semantically meaningful characteristic features for categorizing buildings into one of several categories of architecture.

**The Dataset of European Architectural Styles:** To evaluate the performance while discovering characteristics of architectural styles, we have collected images of 25 different European monuments, which belong to one of five architectural styles. The membership of each monument into architectural style category was validated from Wikipedia. The images for each monument were downloaded from Flickr and Google Image Search. For each monument, its name in English and also in the language spoken in the region where it is located was used as a query while downloading images from the Internet. For example, both "Florence Cathedral" and "Basilica di Santa Maria del Fiore" were used as query to download images for this monument. From the downloaded images, those which provide an external view of the monument were retained and the rest discarded. Table **??** shows the categorization of the dataset into different architectural styles, further broken down into 5 different monuments for each architectural style. There are a total of 6713 high resolution images of size $640 \times 480$ pixels in the dataset. Classification experiments have been performed on the complete dataset.

**Baseline Methods:** To examine the utility of features such as shape, color and appearance for the purpose of categorization of architectural scenes, we experimented with multiple feature detectors and descriptors. To capture color information, color descriptors extracted over local regions and quantized into visual words were implemented. For capturing shape information, we used the HOG descriptor [17] and Geometric Blur [11]. HOG blocks were computed at regular intervals on the images and combined with a Bag of Words approach. Different sizes of the visual codebook (K=1000 and K=4000) were investigated in the case of HOG.

The Shape Context feature was first introduced by Belongie et. al [10] for matching silhouettes. For use in representing architectural images, we first extracted contours of all the images in the database. Small contours (based on the number of points) were rejected. Each contour was represented by the

| Feature | K | Accuracy (in %) |
|---|---|---|
| Shape Context | 1000 | 27.91 |
| Geometric Blur 1 | 4000 | 35.935 |
| Geometric Blur 2 | 4000 | 36.073 |
| HOG | 4000 | 32.90 |
| HOG | 1000 | 35.60 |
| DoG + SIFT | 4000 | 31.31 |
| **PHOW**[63] | **4000** | **46.74** |
| Spatial Pyramid + DSIFT | 84000 | 42.25 |
| **Dense SIFT** | **4000** | **46.22** |

**Table 4.2** Baseline Experiments using multiple feature descriptors and detectors. In Geometric Blur 1 and 2, 4000 and 10,000 features were extracted from each image respectively.

| | Gothic | Korean | Georgian | Islamic |
|---|---|---|---|---|
| Visual Pattern [16] | 93 | 76 | 79 | 77 |
| Affine+SIFT | 93.73 | 96.88 | 94.64 | 78.89 |
| Dense SIFT | 97.27 | 97.50 | 100 | 87.64 |

**Table 4.3** Comparison of our baseline methods and our method of improving classification using Word Mining with the Visual Pattern Discovery method on their dataset for architectural image classification. [16].

Shape Context feature computed over that contour at regularly sampled points. Each image was represented using a histogram of Shape Context descriptors, obtained by assigning Shape Context features to Visual Words (K=4000).

To capture appearance information, we used the SIFT descriptor [39]. Several interest point detectors (Hessian-Affine [42], Difference of Gaussian [39], MSER [41], Dense Sampling) were coupled with SIFT and evaluated. Table 4.2 shows the classification results using the baseline methods. SIFT descriptors computed on a dense grid outperformed the other features and were used for the representation of monument images in our dataset of architectural style categories. The SIFT descriptors were assigned to visual words from a pre-trained vocabulary on a randomly sampled subset of features from the database. The visual vocabulary is obtained using K-means algorithm (K=4000) run 8 times with different random initialization, and keeping the cluster with the lowest energy. SVM classifier was used to perform the classification experiments. We ensure that images of the monument from which the candidate window was generated is absent in the database. This shows how well monuments can be categorized based on architectural styles.

We compare our method of classification with the approach of [16], who also work on the classification of architectural scenes. For this we use their dataset [4] of architectural images. The dataset consists of 423 images, including 111 Gothic images, 156 Korean images, 75 Georgian images, and 81 Islamic images. A 10-fold cross-validation scheme is used. For each fold, 30 images are randomly selected

[4] http://www.cs.ccu.edu.tw/ wtchu/projects/VP/index.html

from each class as the training images, and the remaining is for testing. We show results using multiple methods of image representation. Table 4.3 summarizes the results. We can see that we have created strong baselines, which outperform their classification performance. The average accuracy obtained using Visual Pattern Discovery [16] is $81\%$. Using our baseline of Dense SIFT features, we achieve an average accuracy of $95.6\%$. This demonstrates the superior performance of our method.

### 4.2.1 Mining Pairs of Visual Words Occurrences

We propose a method to mine characteristic pairs of visual words for each architectural category and improve classification. Using higher order feature groups for image classification and object discovery is not new [44, 45, 67, 37, 69]. The utility of doublets and triplets of visual words has been explored for the task of object categorization in [37]. Higher order feature configurations were mined [49] which occurred frequently on instances of a given object class. However, an accurate bounding box enclosing the object was required in the images, from which discriminative feature configurations are mined. We propose a simple, yet effective method to mine doublets of visual words discriminative for each class. Our method works with complete images rather than bounding boxes localizing the buildings in each image.

Using a combination of local features preserves spatial context, which is lost in traditional Bag of Words method. We first create a 2-d histogram of visual word occurrences for each category. The size of the histogram is thus $K \times K$, where K is the number of visual words in the vocabulary. This histogram captures, for each visual word in the codebook, it's co-occurrence with neighboring visual word in images of the same category. This is done by moving a sliding window of fixed size over each image. The 2-d class histogram is updated with the counts of all pairs of visual words that occur in the sliding window.

The counts of bins, across the diagonal, corresponding to the same pair of visual words are summed. Thus, the upper triangular histogram contains complete co-occurrence information for the class.

$$H_c(i,j) = H_c(i,j) + H_c(j,i), i < j \tag{4.1}$$

We now have a 2-d histogram $H_c$ which encapsulates the co-occurrence of visual words for category $c$. The discriminative power of each pair of visual words $d(v_{c,i}, v_{c,j})$ for class $c$ is computed as

$$d(v_{c,i}, v_{c,j}) = |H_c(i,j) - \frac{1}{(NC-1)} \sum_{C=1, C \neq c}^{NC} H_C(i,j)| \tag{4.2}$$

This can be computed for each class in a single step, since the frequency of all pairs of visual words for a given class are stored in a single histogram.

$$dH_c = |dH_c - \frac{1}{(NC-1)} \sum_{C=1, C \neq c}^{NC} H_C| \tag{4.3}$$

| Method | Accuracy |
|---|---|
| Zhang et. al[69] | 78.3% |
| **Word Mining** | 80% |
| QPC[44] | 81.8% |
| QPC+Sel[44] | 80.8% |
| Sgl[44] | 81.7% |
| LPC[44] | 83.9% |

**Table 4.4** Comparison of our method with recent approaches in improving classification using mined word pairs.

### 4.2.2 Utility of Pairs of Visual Words for Image Categorization

The top $P$ visual word pairs with the highest discriminative score are retained for each architectural style category. The initial histogram representation for each image in the database is augmented with an extended histogram of length $P \times NC$, where NC is the number of categories in our dataset. For each image, this extended histogram contains the frequency of the mined pairs in that image. Classification is performed using the augmented image representations.

**Figure 4.4** Top 5 retrieval results for a randomly picked image of San Pietro (Renaissance Architecture). The image outlined in black (left) is the query image. The images outlined in green are the top 5 retrieved images of monuments other than San Pietro, but of the same architectural style. The image outlined in red is a false positive (Romanesque Architecture)

**Comparison with Recent approaches:** Our approach of mining visual word pairs is comparable with recent approaches which use mined higher order features to improve classification accuracy. The effectiveness of $2^{nd}$ and $10^{th}$ order features was explored by Zhang et. al [69]. The correspondence between the same $n^{th}$ order feature across two images is computed. This is done by transforming the local features into offset space. For a visual word $(w, r_1, r_2)$, where $w$ is the visual word, $r_1$ is the location of the word in one image, and $r_2$ is the location of the word in the second image, the position of the word in the offset space is computed as

$$\Delta r = (\Delta x, \Delta y) = (x_1 - x_2, y_1 - y_2) \tag{4.4}$$

Based on the location of words in the offset space, a kernel is computed which is then used for classification using KNN and SVM. We performed experiments on the MSRC v2 dataset[65]. The experimental setup was the same as used in [69]. The size of the sliding window was set to $24 \times 24$ pixels, and $P(= 200)$ most discriminative word pairs were used for each class. Table 4.4 shows how our approach compares with some of the recent approaches. The length of our histogram representation is 5000

| | Gothic | Korean | Georgian | Islamic |
|---|---|---|---|---|
| Visual Pattern [16] | 93 | 76 | 79 | 77 |
| Dense SIFT | 97.27 | 97.50 | 100 | 87.64 |
| **Word Mining** | 98.18 | 99.38 | 98.57 | 97.50 |

**Table 4.5** Comparison of our method of improving classification using Word Pair Mining with the Visual Pattern Discovery method [16].

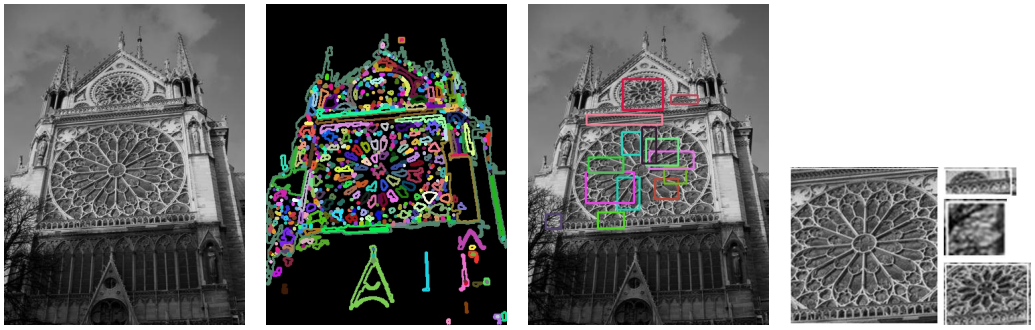
| Method | Accuracy |
|---|---|
| Word Mining | $32.64\%$ |
| SVM | $46.22\%$ |
| **SVM + Word Mining** | $48.25\%$ |

**Table 4.6** Classification Performance on Dataset of European Architectural Styles

$(3200 + 9 \times 200)$. Using this, we obtain better performance than [69], and comparable performance to QPC[44], and QPC + Sel[44], where the number of pairwise feature clusters used is much larger.

**Comparison with Visual Pattern Discovery [16]** [16] propose a method to discover visual patterns in the data and mine sub-graphs of frequently occurring patterns using a graph mining approach. These visual patterns are then used to improve classification performance. We improve our baselines on their dataset using our method of mining discriminative word pairs for each category. The results are summarized in Table 4.5. The size of the sliding window was set to $36 \times 36$ pixels, and $P(= 150)$ most discriminative word pairs were mined for each class. Using our approach of mining discriminative pairs of visual words, we are able to bypass our baseline results using Dense SIFT features. We obtain an average accuracy of $98.41\%$ over four classes.

**Classification Results** The dataset used in [16] is not sub-categorized into monuments. Hence, different images of the same monument can lie in the training and testing sets. Also, the characteristic features to be discovered for a particular architectural style category are present in all the images in the architectural style category for this dataset [16]. We create a much more challenging dataset to evaluate the importance of discovering characteristic features for architectural styles. We ensure that images of the monument from which the candidate window was generated is absent in the database. This ensures that the discovered structures are characteristic to a particular architectural style, and not mere instances of the same building. Table 4.6 shows the classification performance on our dataset. The parameters are the same as used in our previous experiment for comparison with [16]. An interesting observation is the classification accuracy obtained using only the 1000 length histogram of visual word pairs for image representation. The high accuracy ($32.64\%$) shows the utility of doublets of visual words for image categorization. The low classification performance as compared to [16] is mainly due to two reasons - the challenging nature of our dataset, and ensuring that images of the same monument cannot lie in both training and testing sets simultaneously.



(a) Original Image (b) Contour Map (c) Minimum Area Bounding Boxes around medium-sized contours (d) Final seeds obtained after clustering

**Figure 4.5** Generating candidate seeds from an image of the monument - Notre dame de Paris

### 4.2.3 Word Pairs for Image Retrieval

In this section, we evaluate the utility of the mined pairs of visual words for a particular architectural style in retrieving images of the same architectural style category. Similar to image categorization, a Bag-of-Words representation using SIFT keypoints computed on a densely sampled grid are used for image representation. The final representation of the image is a 4000 length L1-normalized histogram. For each architectural style category, 10 samples are randomly selected for one monument for querying. The images of the rest of the monuments are put in the database. A simple nearest neighbor method is used for retrieval. The similarity between two image histograms is computed using the Hellinger Kernel [8], which has shown to give superior performance in image retrieval. To evaluate the retrieval performance, the Mean Average Precision (MAP) is computed over all query samples. The baseline MAP is $22.39\%$, which increases to $22.95\%$ when the histograms are appended with the mined word pairs for each category. Figure 4.4 shows the top 5 retrieved results using pairs of visual word occurrences for a random image from Renaissance Architecture. The improvement in both image categorization and retrieval tasks, however small, is encouraging. We believe that the key to recognition of architectural style categories lies in the larger and semantically meaningful features. We now propose a method to discover such characteristics.

## 4.3 Discovering Semantic Patterns

Starting with a large collection of architectural images, we wish to discover the characteristics that make monuments built according to a particular architectural style distinguishable from other monuments which might follow a different style. This is similar to the problem of Object Category Discovery, which has been previously addressed by several researchers in Computer Vision. Given a large dataset of unlabeled images, the objective is to automatically determine the visually similar categories. Borrowed

from the text mining literature, techniques such as probabilistic Latent Semantic Analysis (pLSA) [32] and Latent Dirichlet Allocation (LDA) [12] for Topic Discovery have been investigated in [56, 57].

We first generate a large set of candidates which serve as potential features for an architectural style category. A possible approach to obtain such a candidate set would be to randomly sample all the images in the database at multiple spatial scales, and exhaustively search through this huge set. However, the number of windows generated in such a manner can be in the order of millions! Our method efficiently filters most of the irrelevant and redundant candidate windows. Inspired by the method used in [53], we extract multiple segmentations from the dataset images guided by segmentation cues. For our database of 2001 images, used for discovery, we obtained a candidate set of 18,127 seeds. The "seeds" are then used for mining clusters of visually similar elements from the database, using a Bag of Words retrieval pipeline. Restricting the size of each cluster to be 100, we still have a huge set of $18,127 \times 100$ patches divided into clusters. The clusters are pruned using a spatial verification step. Based on the number of elements in each cluster, and their SIFT [39] matches with the cluster seed, visually informative clusters are ranked higher. The top few clusters are further refined using LBP features.

A "seed" is a potential architectural feature. We propose a method to generate a reasonable number of seeds guided by segmentation cues. Characteristic features can be of varying shapes and sizes. Candidate windows should be able to completely capture a particular feature in a building. First, edge maps are computed for all images in the dataset using a Canny Edge detector [13]. Closed contours are discovered from the binary edge map using the method of [61]. Each contour consists of the end points of the vertical, horizontal and diagonal line segments it contains. Contours are filtered such that

$$\theta_1 < N < \theta_2 \tag{4.5}$$

where N is the number of end points for the contour. The value of $\theta_1$ and $\theta_2$ has been set to 20 and 500, respectively, in our experiments. This removes most of the noise from the binary image, which may be in the form of people or trees or vehicles. These contours are represented by a bounding box of minimum area enclosing the contour. Any candidate windows generated using these bounding boxes are bound to completely capture a given structure in the image. Often large structures consisting of several smaller sub-structures may not be completely captured. Overlapping bounding boxes are iteratively grouped which results in a single large bounding box for the set of smaller boxes. The clustering procedure is terminated once no overlapping bounding boxes are present. The seeds obtained using the above method are extracted for all images in the dataset, with a slack of 10 pixels in the length and width of the boxes. Figure 4.2.2 gives an overview of the pipeline of generation of candidate windows for one image from the database. The final bounding boxes are segmented from the images. These potential features are now used as seeds for mining similar features across monuments. These features should occur uniformly over all monuments of the same architectural category, and rarely on monuments of different categories.

**Image Representation** Images in our architectural image database, as well as the potential features generated in the previous section, are represented using SIFT [39] descriptors computed at affine-invariant Hessian regions [42]. Randomly sampled SIFT descriptors from a subset of the images in

the database are used to construct a visual vocabulary, which is then used to assign visual word ids to the descriptors. We build an inverted index from the database images which is used to compare vector representations of database images with the potential features. A standard tf-idf weighting scheme is employed.

**Mining Characteristic Features** Each potential feature generated earlier is used as a seed for mining characteristic features. Visually similar characteristics are searched for in the database of images using an inverted index. Due to the large number of generated "seeds", we allow for soft-assignment of database images into more than one clusters. In this phase of clustering, we allow for no more than 100 images per cluster of characteristic feature. The clusters obtained are refined by geometrically verifying the mined characteristics and the initial seeds.

**Spatial Verification:** An affine transformation with 3 degrees of freedom (dof) is fitted between the cluster seed and images falling into the cluster. First, a set of correspondences between SIFT descriptors from the cluster seed and a cluster image are obtained. Similar to [46, 5], hypotheses are generated from only a single pair of correspondences, which has shown to speed up matching and reduce the number of hypotheses generated. While evaluating the generated hypotheses, we allow for large re-projection errors. This is because the appearance of characteristic features can vary significantly across images of different monuments, even though they correspond to the same semantic category of characteristic feature. Images with very few number of inliers obtained from matching with the cluster seed are removed from that cluster.

**Refinement:** The clusters obtained are further refined to remove semantically irrelevant images. For this, we use the LBP feature descriptor. The LBP for a location $(x, y)$ is a string of eight bits, Each bit corresponds to one of the 8-neighbors and is equal to one if it is brighter than the central location. For example, the first bit is one if, and only if,

$$I(x+1, y) > I(x, y) \tag{4.6}$$

The patches are represented using a histogram of LBP features. The distance between the cluster seed and other elements in the cluster is computed. The score of the cluster is computed as a sum of rankings obtained by (a) SIFT matching, and (b) LBP distance.

### 4.3.1 Results

The clusters obtained as a result of mining and pruning are analyzed. Rose windows found across various Gothic monuments by our algorithm have been shown in Figure 4.7. Another characteristic of Gothic monuments are windows with pointed arches as shown in Figure 4.9. Figure 4.10(a) and Figure 4.10(b) show semi-circular windows and arches, which are often found on the facades of monuments of Renaissance architecture. Monuments in Baroque architecture are characterized by a dynamic rhythm of columns (Figure 4.11(a)) and colonnades of two columns at regular intervals (Figure 4.11(b)). All these results have been verified from Wikipedia.

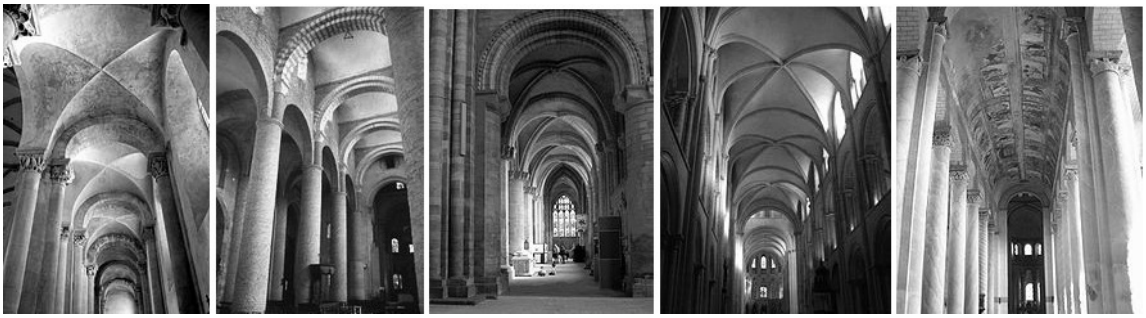


**Figure 4.6** Characteristic Features in Romanesque Architecture which cannot be captured by our method.

## 4.4 Summary

In this work, we explore architectural style categories. We start by identifying the nature of the problem of categorizing buildings into architectural styles categories, and the challenges one might face while solving it. This has been achieved using a set of comprehensive baseline experiments using multiple features. We have evaluated the utility of low-level features in improving the classification performance. The results are encouraging, and motivate us to look for larger and semantically meaningful characteristics. We have proposed a method to identify characteristic features for architectural style categories in an unsupervised fashion. The mined characteristic features are visually informative, and representative of architectural style categories, as verified from Wikipedia. We are, however, limited by the features we employ. There are several characteristic features such as the height of the monument, the plan of the building, or internal features which are difficult to capture. Figure 4.6 shows arched vaults which are commonly found in Romanesque monuments[5], but does not show in our results. Many many more such characteristics is a promising research direction which can be explored by new researchers, who can make the transition from buildings as instances to buildings as categories.

[5]http://en.wikipedia.org/wiki/Romanesque_Architecture

**Figure 4.7** Rose windows as mined by our method are a characteristic feature in Gothic architectural monuments. The cluster seed is from the monument, "Notre Dame de Paris". Similar windows were found in images of other Gothic monument - York Minster, Notre Dame de Chartres and Abbey of St. Denis. The results can be verified from the Wikipedia page `http://en.wikipedia.org/wiki/Rose_window`

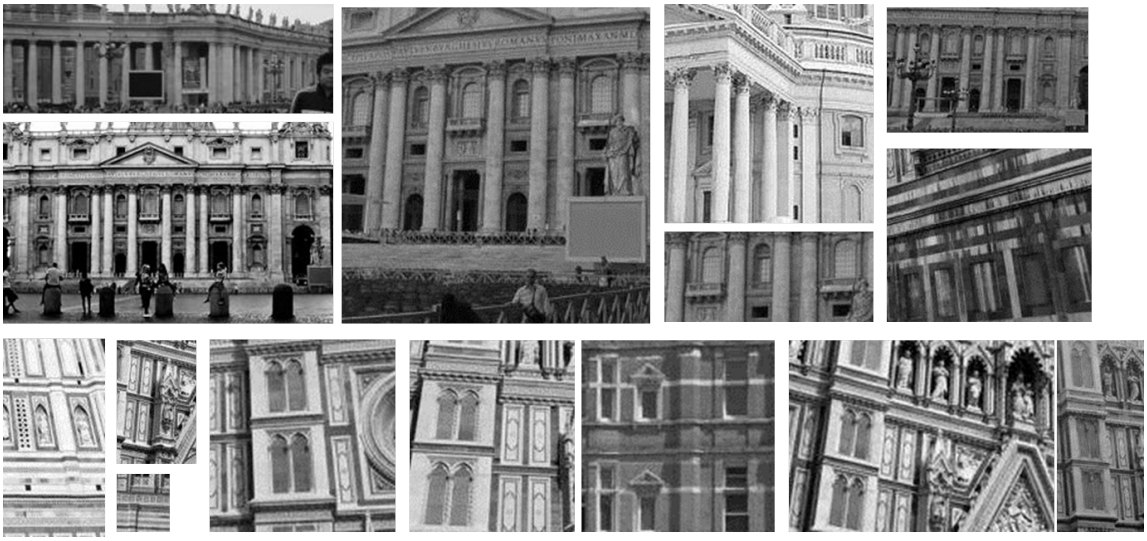
**Figure 4.8**

**Figure 4.9** Pointed arches as discovered by our mining method are a common characteristic in various monuments of the Gothic Architectural Style. Verification of the results can be found at `http://en.wikipedia.org/wiki/Gothic_architecture`

(a) Windows paired with Semi-Circular Arches

(b) Semi-Circular Patterns

**Figure 4.10** Windows with semi-circular arch and semi-circular arches are characteristic of monuments of Renaissance architecture. The figure shows various semi-circular arches discovered by our mining method. The results can be verified from `http://en.wikipedia.org/wiki/Renaissance_architecture`



(a) Dynamic Rhythm of Columns and Pilasters

(b) Colonnades of Free Standing Columns

**Figure 4.11** Monuments in Baroque Architecture are characterized by a dynamic rhythm of columns and pilasters, and two colonnades of free standing columns at regular intervals. The same can be verified from the Wikipedia page of Baroque Architecture `http://en.wikipedia.org/wiki/Baroque_architecture`

*Chapter 5*

# Leveraging Instance Retrieval for Efficient Category Mining

## 5.1 Introduction

The retrieval of instances (same objects) [6, 7, 35, 58, 70] as well as categories (different objects, but same semantics) [18, 21, 40] are both prominent research directions. There are well established methods in both areas which solve the problem to a large extent. *Speed*, *accuracy* and *scalability* to a large database size is of prime importance. Results obtained in instance retrieval are typically more accurate, and obtained much faster due to the relatively easy nature of the problem. Category retrieval is much harder, as visual cues are not sufficient to solve the problem. It involves retrieval at the level of semantics. Category retrieval is often modeled as a classification problem. SVM classifiers are trained for each category [18, 21], or for every exemplar [40, 55, 19]. Training and evaluation of SVM classifiers, however, is computationally expensive. We bridge the gap between instance retrieval and category retrieval, and show how the category retrieval problem can be solved (to an extent) by efficiently adapting instance retrieval. While maintaining accuracy, we are able to greatly speed up the process of retrieval of object categories.

Recently, Gordo *et al.*[28] touched on an important issue in instance retrieval - instance retrieval returns, among the top ranked images, results which are visually similar, but not always semantically consistent with the query. To overcome this issue, they incorporate information from category level labels while searching for instances. We pose a question in the reverse direction - Can instance retrieval be used to benefit retrieval of object categories?

There has been a recent surge of efforts in retrieving instances of the same object from large databases. The challenge is to retrieve accurately, albeit from occlusions, multiple viewpoints and scale variations. The Bag-of-Words (BoW) method [58] has been the mainstay of instance retrieval techniques for many years. It was initially used to represent images as a histogram of visual words, obtained by vector quantizing SIFT descriptors [38] computed at interest points. Coupled with an inverted index, enabling fast indexing and searching, and tf-idf weighting to downscale the contribution of frequently occurring visual words, this method has become the popular instance retrieval pipeline.

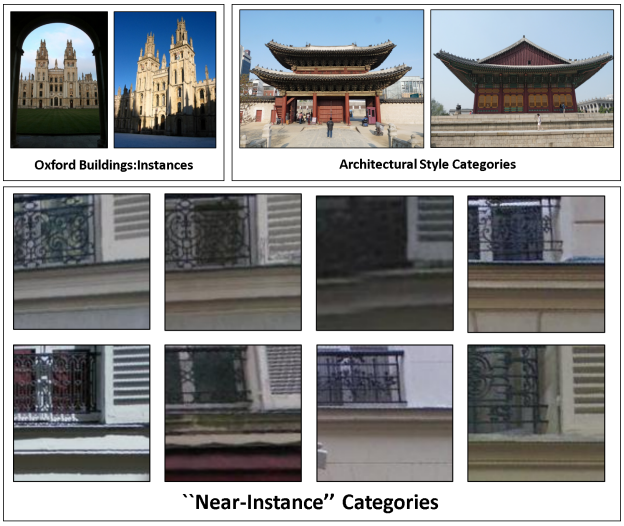



**Figure 5.1** On one hand, buildings have been explored as instances in the Oxford buildings dataset[47] (top-left), while as categories of architecture on the other [16] (top-right). Several object categories, such as those shown here (bottom) are much nearer to instances, despite being obtained from different buildings. We are interested in such near-instance categories.

To ensure spatial consistency between the query and the retrieved results obtained by the order-less Bag-of-Words method, geometric verification was introduced as a post processing step [58]. Spatial consistency can be enforced loosely by ensuring that matching visual words are from a small neighborhood in the retrieved image, or strictly by estimating a fundamental matrix from point correspondences using methods such as RANSAC [24]. The computational complexity of the geometric verification step motivated research in incorporating it into the retrieval pipeline [70, 35] itself, rather than being used as a post processing step.

The instance retrieval pipeline is both fast and robust. It scales well with database size and visual vocabulary size. Instance of the same object can be retrieved in the order of milliseconds from huge data sets containing many thousand images. The high quality of results obtained for retrieving object instances has lead to several interesting applications [2, 6, 7].

While a plethora of work has been reported in recent past on retrieving similar instances of an object, the problem of image-category search, however, did not receive much attention as a retrieval problem. Rather, the latter has been well received as a learning-based classification problem. [18, 21]. Since the location and spatial extent of the object is unknown, such methods involve an exhaustive evaluation of all possible windows, at multiple spatial scales. This process is computationally expensive. Solutions proposed for solving an unsupervised variant of this problem employ methods borrowed from the data mining community [23, 49, 51, 52, 66]. Even though easily scalable to millions of transactions, adapting mining methods for solving computer vision tasks faces several challenges. The uncertainty in feature representations in images makes it hard for popular itemset mining schemes, such as Apriori [4] and FP-Growth [29] to be directly applied.

State-of-the-art methods [18, 21, 40] in object category retrieval/mining learn a SVM classifier for a set of labeled objects. This classifier (or filter) is applied to all possible patches, in a sliding window fashion, at multiple scales. This number is in the order of millions! The SVM score determines the presence or absence of the object in the image patch. Despite being accurate, classifier based methods are computationally expensive when retrieving object categories from a large data set, which is primarily due to two reasons. The training time of an SVM for a considerable set of positive and negative exemplars is high. Previous works [18, 21] have proposed mining of hard negatives from millions of negatives exemplars. This improves classification accuracy, but at the cost of computation time due to multiple rounds of SVM re-training. [40] proposed a method of learning a separate classifier for each positive exemplar in the given set. This compensates the problem of overly-generic detectors being learnt, and each positive exemplar is given the appropriate amount of attention. However, the number of classifiers increase manifold, leading to an increase in testing time.

We propose an instance-retrieval based solution for searching object categories in large databases. We discuss what are challenges faced when leveraging instance level techniques for solving category retrieval, and efficiently adapt the instance retrieval pipeline. Using our method, we are able to perform object category search which is 1000 times faster than an exemplar based SVM classifier. We also perform unsupervised mining using our method, and discover several near-instance object categories automatically. We first talk about the popular instance retrieval pipeline in the next section, which is the base of our approach.

## 5.2 Efficient Category Retrieval

We leverage the existing instance-retrieval pipeline to solve the problem of category retrieval. We propose a simple solution to re-rank the retrieved list of results using the HoG feature descriptor [18], and obtain better results on several object categories. We evaluate our approach on a variety of object categories. Our method works well for a subset of these categories. These categories, which we term as near-instance categories, exhibit several common characteristics, such as relatively less intra-class variation in visual appearance. We are also able to surpass the time complexity barrier, which is a problem when using an SVM based classifier, and retrieve objects 1000 times faster than a SVM-based category classifier. While the instance retrieval pipeline accompanied by spatial verification is able to obtain impressive results for retrieval of same objects, it is not able to do the same for category retrieval. The typical vocabulary size used in category retrieval/classification is 4000. Thus, visual words are better able to capture the variation across categories. On the other hand, in instance retrieval we are interested in exact matches, hence using a much larger vocabulary size (1 million visual words). We choose a vocabulary size of 10,000 visual words, which is more suitable for representing "near instance" categories.

**HoG Post-processing:** Retrieval using SIFT is dominated by low level features. [19]. Histogram of Oriented Gradients (HoG) [18] has shown to efficiently capture shape information in large image

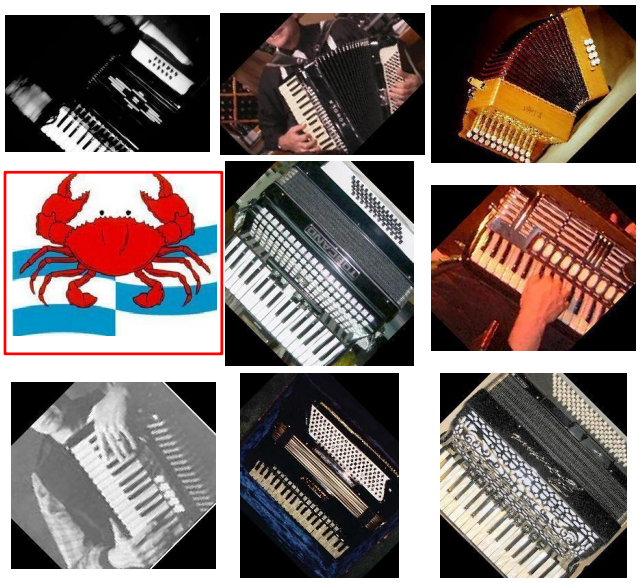


**Figure 5.2** Top retrieved images (from top-left, in row-major form) for query image of class "Accordion". The relative similarity in viewpoint and appearance as compared to other classes makes this category "near-instances". The image outlined in red is a false positive.

patches. We propose a HoG-based post processing method to improve the quality of retrieved results. HoG feature descriptors, with a block spacing stride of 8 pixels, are obtained for the query as well as retrieved images. The retrieved images are re-sized to the size of the query image, ensuring consistency in the dimensions of HoG vector representations. The descriptors for each image are concatenated into a 1-dimensional vector. The Euclidean distance between the HoG vector representations of the query sample and that of the retrieved images is computed. Two retrieved images are compared as

$$\begin{cases} Rank(I_1) < Rank(I_2) & \text{if } d(I_1, q) < d(I_2, q) \\ Rank(I_1) > Rank(I_2) & \text{if } d(I_1, q) > d(I_2, q) \end{cases}$$

We use our method to retrieve near-instance object categories. We propose that our method of HoG based reranking be employed as a post processing step to geometric verification. Accurate bounding boxes around the retrieved object, estimated from the geometric matches between SIFT key-points can easily be obtained. Comparison of vector representations of objects contained in the bounding box compensates for the translation invariant nature of the HoG descriptor. We test our method on the Caltech Data set of 101 Object Categories. It consists of a variety of object categories. Objects in each category are aligned across the images, which makes it an ideal setting for the evaluation of our HoG Post-processing step. We compare against multiple baselines. In the first baseline, we use the simple instance retrieval pipeline (without spatial verification) to retrieve object categories. As a second baseline, we train a linear SVM classifier for each query exemplar. HoG vector representation of the query exemplar is used as a positive training instance. Thousand negative exemplars are obtained by randomly selecting 10 samples from each class in the dataset (except the class of the query exemplar).

For each database image, a score is obtained for every possible window containing the same number of cells as the HoG template of the positive exemplar. The image score is

$$Score(I) = max(I_{m,n}) \tag{5.1}$$

where $I_{m,n}$ is the set of all possible HoG windows.

In a separate experiment, we also use the images retrieved by our approach for training a linear SVM. The top 20 retrieved images are used as positive samples, instead of using a single positive exemplar. The rest of the method is similar to the second baseline.

The Caltech 101 data set [20] consists of objects belonging to 101 categories, with the number of images per category ranging from 40 to 800. It contains a total of 9146 images. The dimensions of each image are roughly $300 \times 200$ pixels. SIFT descriptors are computed for each image in the database at interest points obtained using a Difference of Gaussian (DoG) detector. A random subset of 100,000 SIFT descriptors obtained from the database images was used to create a vocabulary. The vocabulary size is chosen to be considerable large (10,000). This ensures that only significantly similar features are captured by each visual word. 5 images were randomly selected from each class to create the set of query images.

Two measures have been used to report retrieval accuracy. (a) The precision obtained for the top 10 retrieved samples was computed. The average over 5 randomly chosen queries, known as *Mean Precision at 10* has been reported for each category. (b) The average precision was computed for the complete rank list. The average over 5 randomly chosen queries *Mean Average Precision* has been reported for each category.

Based on the results of our experiments, we are successfully able to divide the 101 Object Categories of the Caltech dataset into 3 sets. Images in each of the three sets exhibit different properties, based on which we infer how easy/tough it is to retrieve from them. The first set of categories (Table 5.1) are those which are *near-instances*. Using an Instance Retrieval based method accompanied by a HoG-based post processing step gives much better performance, both in terms of precision at 10 as well as average precision, than an exemplar SVM based method, as well as the standard instance retrieval pipeline. Figure 5.2 shows the top retrieved results for a query of the class "Accordion". Even though the images contain different models, there is similarity in appearance and viewpoint across all images, which makes it an ideal candidate for a "near-instances" object category.

The second set of categories are *neutral* categories (Table 5.2). From observing the images belonging to these classes, we inferred that even though there is some similarity in visual appearance across images, it is not sufficient for instance retrieval to work. This observation is supplemented by the similar retrieval performance obtained by our method as well as 1-Exemplar SVM for these categories. The third set are *tough* categories (Table 5.3). Images of objects in these classes exhibit high intra-class variance, due to which standard category retrieval methods such as SVM often outperform our method of retrieving near-instances.

**Time Analysis:** To evaluate how fast instance retrieval really is, we implemented an inverted-index based search system using Lucene [3]. Lucene is a free open-source software written in Java for infor-

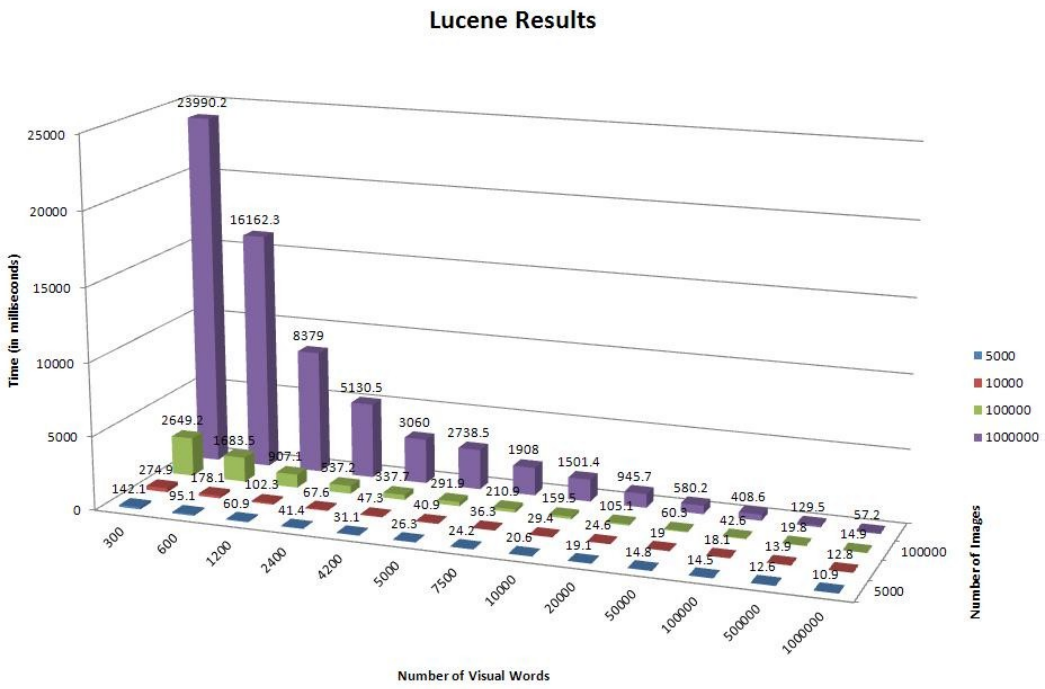



**Figure 5.3** Plot showing how time taken to retrieve from an inverted index scales with (a) Number of Visual Words (X-axis), and (b) Number of Images (Color-axis)

mation retrieval. Using a vocabulary size of 10,000 visual words, the time taken for a single retrieval query is 29.41 milliseconds. Figure 5.3 visualizes the scaling of time taken to retrieve from an inverted index with large number of visual words, and images.

**Conclusion:** Several near-instance object categories exist. Images in these categories exhibit similar visual appearance, despite containing different objects. SVM-based methods, despite giving good results, are computationally expensive. Using an *instance-based* solution for retrieving from such categories is both fast, and efficient.

## 5.3 Efficient Category Mining

There have been many attempts in the past at adapting well-established data-mining techniques for unsupervised mining in images. Quack et al [52] mine clusters containing similar image content from geo-tagged imagery obtained from Flickr. The mined clusters are then classified into objects/events, and text labels, obtained from Wikipedia articles, are associated with each cluster. Sets of discriminative patches, termed as Grouplets, were mined for modeling the interactions between humans and objects in [66]. Standard mining methods in computer vision [23, 49, 51, 52, 66] involve building a database of

(a) Electric Guitar - Tough Category

(b) Wheelchair - Neutral Category

| | Mean Precision At 10 | | | Mean Average Precision | |
|---|---|---|---|---|---|
| **Class** | **IR** | **IR + HoG** | **1E-SVM** | **IR** | **1E-SVM** |
| Faces | 0.488889 | 0.660000 | 0.303704 | 0.436336 | 0.143075 |
| Leopards | 0.800000 | 0.900000 | 0.533333 | 0.305879 | 0.167677 |
| Motorbikes | 0.977778 | 1.000000 | 0.533333 | 0.462378 | 0.309193 |
| accordion | 0.600000 | 0.820000 | 0.511111 | 0.179096 | 0.020549 |
| airplanes | 0.666667 | 0.860000 | 0.488889 | 0.265543 | 0.136673 |
| barrel | 0.177778 | 0.160000 | 0.111111 | 0.033934 | 0.010027 |
| brain | 0.288889 | 0.380000 | 0.111111 | 0.112103 | 0.009324 |
| dollar bill | 0.600000 | 0.720000 | 0.437037 | 0.235146 | 0.009601 |
| garfield | 0.377778 | 0.380000 | 0.288889 | 0.139086 | 0.098127 |
| grand piano | 0.466667 | 0.520000 | 0.362963 | 0.133576 | 0.013609 |
| soccer ball | 0.533333 | 0.460000 | 0.288889 | 0.171098 | 0.015442 |
| starfish | 0.444444 | 0.320000 | 0.333333 | 0.097817 | 0.034934 |
| stop sign | 0.822222 | 0.820000 | 0.251852 | 0.720323 | 0.021860 |
| sunflower | 0.377778 | 0.440000 | 0.281481 | 0.150520 | 0.011299 |
| tick | 0.377778 | 0.300000 | 0.244444 | 0.140898 | 0.013260 |
| windsor chair | 0.622222 | 0.660000 | 0.362963 | 0.387337 | 0.161913 |
| yin yang | 0.622222 | 0.660000 | 0.333333 | 0.245286 | 0.013444 |

**Table 5.1** Comparison of Various Methods for Object Category Retrieval on Near-Instance Categories. IR = Instance Retrieval, IR + HoG = Instance Retrieval + HoG-based Re-ranking, 1E-SVM = One Exemplar SVM

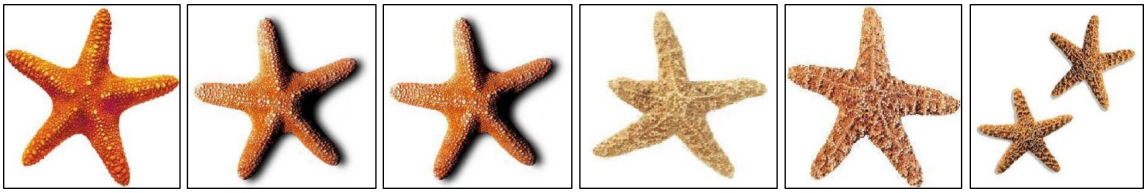
(c) Airplanes

(d) Motorbikes

(e) Faces

(f) Windsor Chair

(g) Dollar Bill



(h) Starfish


(i) Stop Sign

**Figure 5.4** Top retrieved results for several near-instance categories.

| | Mean Precision At 10 | | | Mean Average Precision | |
|---|---|---|---|---|---|
| **Class** | **IR** | **IR + HoG** | **1E-SVM** | **IR** | **1E-SVM** |
| Faces easy | 0.333333 | 0.480000 | 0.303704 | 0.135929 | 0.143075 |
| anchor | 0.222222 | 0.180000 | 0.177778 | 0.051034 | 0.020461 |
| ant | 0.111111 | 0.120000 | 0.111111 | 0.030102 | 0.008435 |
| bass | 0.111111 | 0.100000 | 0.122222 | 0.025482 | 0.013824 |
| beaver | 0.111111 | 0.100000 | 0.111111 | 0.029775 | 0.013118 |
| binocular | 0.133333 | 0.120000 | 0.122222 | 0.037531 | 0.014704 |
| bonsai | 0.177778 | 0.180000 | 0.133333 | 0.043176 | 0.042940 |
| brontosaurus | 0.111111 | 0.120000 | 0.111111 | 0.031374 | 0.028261 |
| butterfly | 0.133333 | 0.140000 | 0.133333 | 0.028946 | 0.010393 |
| camera | 0.133333 | 0.140000 | 0.111111 | 0.029636 | 0.009470 |
| cannon | 0.111111 | 0.100000 | 0.111111 | 0.030871 | 0.013974 |
| ceiling fan | 0.155556 | 0.160000 | 0.133333 | 0.032675 | 0.010509 |
| cellphone | 0.111111 | 0.140000 | 0.133333 | 0.029367 | 0.016849 |
| euphonium | 0.222222 | 0.240000 | 0.251852 | 0.047642 | 0.022716 |
| pagoda | 0.266667 | 0.240000 | 0.259259 | 0.083303 | 0.011967 |
| scissors | 0.244444 | 0.180000 | 0.244444 | 0.069535 | 0.025773 |
| snoopy | 0.244444 | 0.280000 | 0.274074 | 0.108894 | 0.026255 |
| stegosaurus | 0.244444 | 0.280000 | 0.274074 | 0.050942 | 0.013389 |
| strawberry | 0.222222 | 0.220000 | 0.237037 | 0.062800 | 0.013172 |
| umbrella | 0.200000 | 0.140000 | 0.244444 | 0.040433 | 0.008227 |
| water lilly | 0.244444 | 0.220000 | 0.237037 | 0.075066 | 0.012747 |

**Table 5.2** Comparison of Various Methods for Object Category Retrieval on Neutral categories. IR = Instance Retrieval, IR + HoG = Instance Retrieval + HoG-based Re-ranking, 1E-SVM = One Exemplar SVM

| | Mean Precision At 10 | | | Mean Average Precision | |
|---|---|---|---|---|---|
| **Class** | **IR** | **IR + HoG** | **1E-SVM** | **IR** | **1E-SVM** |
| buddha | 0.111111 | 0.100000 | 0.166667 | 0.022950 | 0.019182 |
| car side | 0.111111 | 0.140000 | 0.411111 | 0.038791 | 0.307318 |
| chair | 0.111111 | 0.100000 | 0.244444 | 0.026110 | 0.053971 |
| chandelier | 0.133333 | 0.120000 | 0.237037 | 0.028906 | 0.011793 |
| cougar body | 0.155556 | 0.140000 | 0.244444 | 0.039822 | 0.021603 |
| cougar face | 0.133333 | 0.140000 | 0.244444 | 0.039136 | 0.021603 |
| crab | 0.111111 | 0.100000 | 0.244444 | 0.027032 | 0.021603 |
| crayfish | 0.111111 | 0.100000 | 0.244444 | 0.023120 | 0.021603 |
| crocodile | 0.155556 | 0.120000 | 0.244444 | 0.035079 | 0.021603 |
| crocodile head | 0.111111 | 0.100000 | 0.244444 | 0.028146 | 0.021603 |
| dolphin | 0.155556 | 0.120000 | 0.437037 | 0.033813 | 0.009601 |
| dragonfly | 0.177778 | 0.180000 | 0.311111 | 0.034388 | 0.055220 |
| electric guitar | 0.111111 | 0.100000 | 0.281481 | 0.021129 | 0.033177 |
| elephant | 0.111111 | 0.100000 | 0.237037 | 0.027111 | 0.016869 |
| emu | 0.111111 | 0.100000 | 0.259259 | 0.033987 | 0.018108 |
| ewer | 0.111111 | 0.120000 | 0.244444 | 0.023285 | 0.012600 |
| flamingo | 0.133333 | 0.120000 | 0.244444 | 0.032325 | 0.011804 |
| flamingo head | 0.133333 | 0.120000 | 0.244444 | 0.036571 | 0.011804 |
| gerenuk | 0.133333 | 0.120000 | 0.251852 | 0.050801 | 0.020210 |
| gramophone | 0.155556 | 0.140000 | 0.259259 | 0.032295 | 0.011431 |
| hawksbill | 0.133333 | 0.140000 | 0.244444 | 0.032115 | 0.028825 |
| headphone | 0.133333 | 0.160000 | 0.259259 | 0.044063 | 0.011630 |
| hedgehog | 0.133333 | 0.120000 | 0.274074 | 0.039610 | 0.017521 |
| helicopter | 0.111111 | 0.120000 | 0.266667 | 0.025053 | 0.007548 |
| ibis | 0.111111 | 0.120000 | 0.244444 | 0.028375 | 0.011877 |
| inline skate | 0.155556 | 0.140000 | 0.355556 | 0.044505 | 0.013818 |
| joshua tree | 0.133333 | 0.120000 | 0.244444 | 0.042296 | 0.011627 |
| kangaroo | 0.111111 | 0.100000 | 0.259259 | 0.027485 | 0.017769 |
| ketch | 0.222222 | 0.220000 | 0.311111 | 0.045442 | 0.076220 |
| lamp | 0.111111 | 0.100000 | 0.237037 | 0.025706 | 0.009432 |
| laptop | 0.155556 | 0.220000 | 0.296296 | 0.046374 | 0.007819 |
| llama | 0.133333 | 0.120000 | 0.244444 | 0.031471 | 0.019477 |
| lobster | 0.111111 | 0.100000 | 0.237037 | 0.030422 | 0.011494 |
| lotus | 0.111111 | 0.100000 | 0.251852 | 0.026887 | 0.009473 |
| mandolin | 0.133333 | 0.120000 | 0.244444 | 0.029481 | 0.029217 |
| mayfly | 0.111111 | 0.100000 | 0.237037 | 0.030512 | 0.011909 |
| menorah | 0.133333 | 0.120000 | 0.244444 | 0.058681 | 0.009334 |
| metronome | 0.155556 | 0.160000 | 0.318519 | 0.055514 | 0.086379 |
| minaret | 0.200000 | 0.240000 | 0.400000 | 0.056404 | 0.146049 |
| nautilus | 0.177778 | 0.180000 | 0.251852 | 0.041649 | 0.010437 |
| octopus | 0.133333 | 0.120000 | 0.251852 | 0.038720 | 0.013495 |
| okapi | 0.111111 | 0.120000 | 0.244444 | 0.033006 | 0.016690 |
| panda | 0.111111 | 0.140000 | 0.237037 | 0.041265 | 0.011686 |

**Table 5.3** Comparison of Various Methods for Object Category Retrieval on Tough categories. IR = Instance Retrieval, IR + HoG = Instance Retrieval + HoG-based Re-ranking, 1E-SVM = One Exemplar SVM

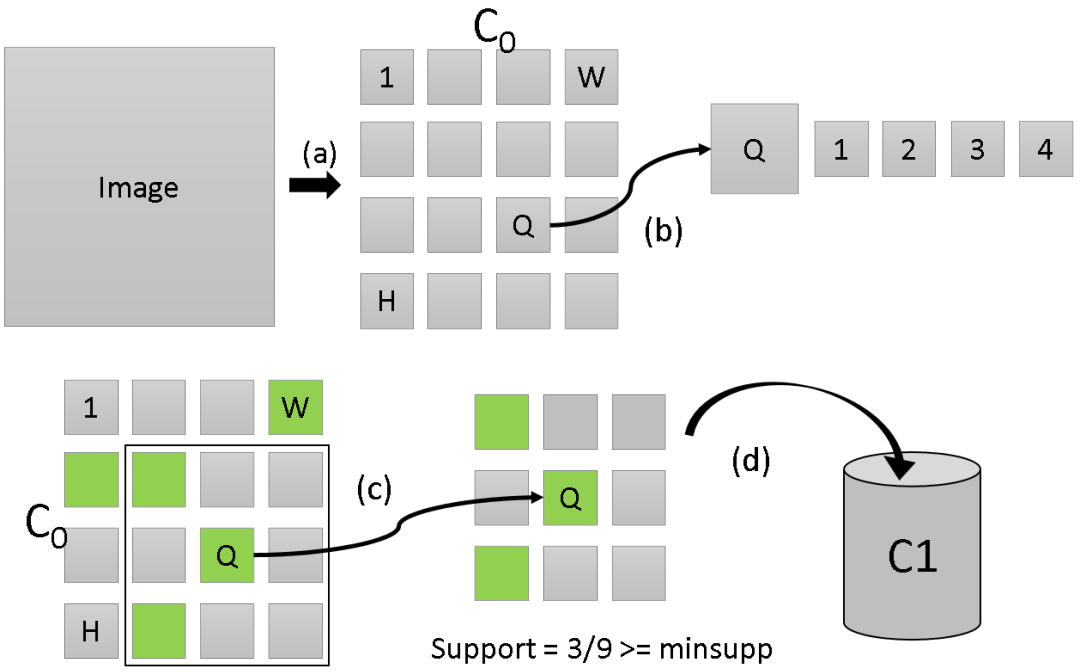



**Figure 5.5** Our mining pipeline: (a) Image is divided into small, square patches. (b) Each patch is used to retrieve similar patches from the database. (c) Patch with high Goodness Score (marked in green), are grown by grouping with nearby patches. (d) Grouped patches which contain atleast *minsupp* good patches are used for further mining/retrieval.

transactions. Feature neighborhoods consisting of multiple visual words are modeled as a transaction. [51] create a transaction from $k$ nearest visual words for each selected word $v_c$. [49] use the scale of the central visual word to define the size of the neighborhood. Each transaction contains multiple items. To compensate for the uncertainty in feature representations in images, features are quantized and assigned visual word identifiers. Each visual word is modeled as an item in a transaction. This can lead to noisy or missing items in a transaction created from a feature neighborhoods.

Due to the unknown location and spatial extent of objects in images, an almost infinite number of transactions need to be enumerated and evaluated for mining of objects. [23] create transactions from local histograms. They propose a relevance criterion to filter through the large set of transactions, and evaluate the utility of the mined Frequent Local Histograms (FLH) for image classification. Topic models have been investigated for the purpose of unsupervised discovery of objects [56, 54, 57]. Classifier based methods apply a SVM classifier to all possible transactions, which is computationally expensive. We propose an instance-based solution for solving the problem of discovering near-instance object categories in a large dataset. We progressively evaluate patches (or transactions) at increasing spatial scales, all the while rejecting uninteresting transactions at every step. As a result, we are able to quickly discover a set of semantically meaningful near-instance object categories.

The given image is first divided into square patches of fixed size of $25 \times 25$ pixels. This starting set of patches constitutes the initial candidate set $L_0$. Each candidate patch is evaluated and a score assigned to it which signifies the goodness of the patch as belonging to a larger, semantically meaningful and frequently occurring near-instance category. The top scoring patches from the candidate set $L_k$, based on the "Goodness" score, are used to create the candidate set $L_{k+1}$, where $k$ denotes the level of grouping. The patches used to create the subsequent candidate sets are increased in size by incorporating a larger region around the current patch. Figure 5.5 gives a step by step overview of our method. We now talk about the algorithm in detail.

**Patch Evaluation:** Each patch in the candidate set is represented using a histogram of visual words. This patch is now used as a query to retrieve visually similar patches across the image database. The top few retrieved patches are re-ranked using a spatial verification step. We fit an affine transformation with 3 degrees of freedom (dof) between the query and retrieved patches. Similar to [6, 47], hypotheses are generated from single point correspondences, which speeds up matching. To accommodate matches on near-instance categories, we allow large-re-projection errors. Due to the small size of the query patch, the number of matches to be evaluated is typically small. The retrieved list of patches is characterized by many false positives. This is mainly due to (a) using a reasonably large vocabulary size for matching object categories (not instances), and (b) small size of the query patch, which results in very small number of matches from spatial verification. We propose a goodness score to measure the quality of the retrieved patches obtained in the cluster. The Goodness Score of a patch is computed as

$$GScore(I_p) = \left( \frac{\sum_{i=1}^{N} d(HoG_p, HoG_i)}{N * A} \right)^{-1} \tag{5.2}$$

where $d(HoG_p, HoG_i)$ is the Euclidean distance between the HoG vector representations of the query patch $I_p$ and the $i^{th}$ retrieved patch, $N$ is the number of patches retrieved, and $A$ is the area of the query patch. The patches are re-sized to the size of the query before the HoG descriptors are computed, to ensure consistency in dimension across HoG vector representations. A larger value of N enforces a much stricter goodness constraint on each patch, at the cost of increased computation in calculating the distances between the HoG vector representations. Normalizing by the area enables fair comparison between the scores assigned to patches of varying sizes.

**Patch Growing:** The patches obtained from candidate set $L_k$ with a high Goodness Score are grown to create a similar set $L_{k+1}$ for next level of evaluation. A simple solution for growing the patch can be to select 8-connected neighboring patches of the current patch. A fundamental problem with this is that some parts of the new patch might be uninteresting, and may not have any semantic association with the object which, if present, we are trying to discover. Another method of selecting useful patches is non-maximum suppression (in terms of Goodness Score), where a larger image region around each local maximum is selected. At the $k^{th}$ level of grouping, we select $(2k+1) \times (2k+1)$ square patches,


**Figure 5.6** Figure showing improvement in cluster purity after HoG Re-ranking for a particular patch (House Numbers) at Level 2 of mining. The image outlined in black is the initial patch. Two false positives (outlined in red) appeared after mining (Row 1). These were rejected after re-ranking using HoG (Row 2).

where the size of each patch is $25 \times 25$ pixels. The support of this new patch $(Patch_k)$ of increased size is computed as

$$Supp(Patch_k) = \frac{M}{(2k+1) * (2k+1)} \tag{5.3}$$

where M is the number of patches in the image region which are among the top ranked ones based on Goodness Score. Regions with support value greater than a minimum threshold *minsupp* are used to create the candidate set $L_{k+1}$ for the next level of evaluation.

**Mining Near-Instances:** After multiple levels of refining, we are left with few, reasonably large sized patches. Similar to the method of patch evaluation, these reasonably large patches are used to query from the database. We retrieve the top 100 results for each query region. A HoG based post-processing step is employed to refine the clusters of near-instance object categories. The final rank of a retrieved patch is computed as

$$Rank = max(Rank_{SIFT}, Rank_{HoG}) \tag{5.4}$$

where $Rank_{SIFT}$ is the original rank in retrieved list, and $Rank_{HoG}$ is computed based on the Euclidean distance between the HoG templates of the query and retrieved image regions. A lower value of $Rank$ signifies that the result appears at the top of the rank list.

Our HoG-based re-ranking method suppresses false positives by pushing them away from the query, while retaining true positives in the top results. This method of pushing away false positives retains the ranking information obtained by SIFT matching, which is important for retrieval of near-instance object categories. Figure 5.6 visualizes improvement in purity of one particular mined cluster (House Numbers) after rejecting false positives using our method.

(a) Text Signs (b) Lamps

(c) Windows (d) Grills

**Figure 5.7** Various near-instance Object Categories which were automatically discovered by our mining method

| Task | Time Taken (in sec.) |
|---|---|
| Similarity Search | 0.797 s |
| Spatial Verification | 0.324 s |
| HoG Score Computation | 0.468 s |

**Table 5.4** Time taken to evaluate a single patch of size $25 \times 25$ pixels.

### 5.3.1 Computational Complexity

Our method is fast. Table 5.4 gives the time taken by each module of our approach for evaluating a single patch. The implementation has been done in MATLAB on a standard PC with 4 GB RAM. Evaluating multiple patches can easily be done in parallel since these are independent tasks. An exact time comparison with previous works for category retrieval and mining may not be possible due to the different settings (data sets, features) employed earlier. We provide a theoretical evaluation of the time complexity of our approach with two popular approaches in computer vision - Frequent Item set Mining, and Exemplar SVM.

**Frequent Item set Mining** Consider an image database of size $|D|$. If the average number of visual words detected in an image is $N$, and a transaction is created for each visual word, the total number of transactions used to create a database for a standard mining algorithm is $N \times |D|$. For a vocabulary size $K$, there are typically $2^K$ possible item sets to be evaluated. If spatial information about a feature neighborhood is incorporated, similar to [51], the number of possible item sets further increases to $2^{4*K}$. Considering we want to mine reasonably large sized patches, one can enumerate all item sets and check

if they are frequent in time $O(2^i \times N \times |D|)$ [68], where $i$ is the maximum item set length (approximate number of visual words in our case, depending on the scale of objects which we are trying to mine.

**Exemplar SVM** Mining/Retrieval of Object Categories using Exemplar SVM comprises of two stages - training and testing. Training a linear SVM involves maximizing the margin between the single positive exemplar and millions of negative exemplars. For both primal and dual optimization, the complexity is $O(max(n,d)min(n,d)^2)$ [14], where $n$ is the number of training instances (1 Positive + Many Negatives), and $d$ is the dimension of representation of an exemplar. Both retrieval as well as mining are unsupervised methods, and require no time for training, as opposed to an SVM classifier. SVM testing scales linearly with the number of positive instances (but not the negatives), since a separate classifier is learnt for each positive exemplar. For an image (of dimension $H \times W$), applying a classifier (at say $S$ spatial scales) takes $O(H \times W)$ computation (since $S << H \times W$). We can evaluate all $E$ Exemplar-SVMs over $|D|$ database images in $O(E \times |D| \times H \times W)$.

### 5.3.2 Results

We perform both quantitative and qualitative evaluation of our method. For the discovery of near-instance object categories, we use the data set of Google Street View Images as provided by [19]. The images were obtained for 12 cities: : Paris, London, Prague, Barcelona, Milan, New York, Boston, Philadelphia, San Francisco, San Paulo, Mexico City, and Tokyo. We downloaded 25,278 images from Google Street View, the dimensions of each image being $936 \times 537$ pixels. This gives us $38 \times 22 = 836$ patches per image, when the starting patch size is $25 \times 25$ pixels. A starting set of 100 seed images was used for the purpose of discovery.

**Image Representation:** SIFT descriptors [38] are computed at affine-invariant Hessian regions [43]. Each SIFT descriptor is assigned a visual word id from a visual codebook. For image representation, we use a vocabulary trained on the Oxford Buildings data set [47] comprising of 100,000 visual words. An inverted index is built from the database images, which enables fast searching. A standard tf-idf weighting scheme is used, which suppresses the contribution of frequently occurring visual words. The Hellinger kernel is used for comparison of image vector representations, which has shown superiority over Euclidean distance in texture classification and image categorization tasks [9].

#### 5.3.2.1 Quantitative Evaluation

Here, we show a quantitative comparison of our mining method with various other existing methods in the literature for the purpose of mining near-instance object categories.

**Exemplar SVM:** A linear SVM is learnt for a single positive exemplar, and all possible windows from database images (other than query image) as the negative exemplars. We perform 10 iterations of retraining, and 5 iterations of mining hard negatives in each iteration of re-training. 100 hard negatives are mined in each iteration (of re-training) and added to the negative set. The PEGASOS SVM solver in the VLFeat library is used for training. The bias multiplier is set to 100, and the regularization parameter

| Instance based Solution | Time (in sec) |
|---|---|
| Inverted Index Creation (offline) | 16.088 |
| Similarity Search | 0.012 |
| Geometric Verification | 0.411 |
| HoG Reranking | 0.070 |
| **Total (in mins)** | **0.0082** |

**Table 5.5** Breakdown of Computational Complexity using Instance Retrieval

| Exemplar SVM | Time (in min) |
|---|---|
| Training (offline) | 22.66 |
| Testing | 26.66 |
| **Total (in mins)** | **49.32** |

**Table 5.6** Training and Testing time taken by Exemplar SVM method when search for an object in a small image database of 200 images.

$\lambda = 100/N$, where $N$ is the number of training samples. For training, the classifier is run on a subset of 200 database images, at 4 spatial scales. We also perform retrieval on this subset using the same positive exemplar, followed by the HoG post-processing method. Table 5.5 summarizes the computation times for both methods.

**Frequent Pattern Mining [51]:** We perform Frequent Pattern Mining. Similar to [51], a neighborhood is built around each sampled feature location $R_c$. The scale of the central feature $R_c$ is used to define the size of the neighborhood. Each neighborhood is split into $Q$ tiles, and an activation vector is created for each tile, denoting the visual words contained in it. The $Q$ activation vectors (1 for each tile) are concatenated to form a neighborhood descriptor, and effectively listed as transaction. The value of $Q$ has been set to 3 in our experimental comparison. The Apriori Algorithm [4] is used to mine frequently occurring patches in the given set of image transactions. Table 5.7 shows the time taken by this approach. Since a transaction is created for every feature descriptor, there are typically a large number of transactions generated from even a small set of images, leading to high database generation time. Performing Frequent Itemset Mining using Apriori becomes impossible due to the large number of distinct items (100,000), which requires a total of $2^{100,000}$ possible itemsets to be evaluated.

**Summary:** Experiments on time comparison demonstrate the superficial nature of our method in terms of speed as compared to both supervised exemplar-SVM based technique as well as unsupervised Frequent Itemset Mining in mining near-instance object categories. Both offline time as well as online time is reduced.

**Qualitative Evaluation:** We perform mining upto 3 levels. The top 20 results were retrieved for each patch upto the penultimate mining level. Top 100 results were obtained for visualization at the last level. From the set of 100 seed images, we discovered several different concepts (at multiple levels), which fall into the category of near-instance objects. Figure 5.9 showcases examples of few near-instance object categories discovered by our method. The include balconies, street lamps, windows, and banners

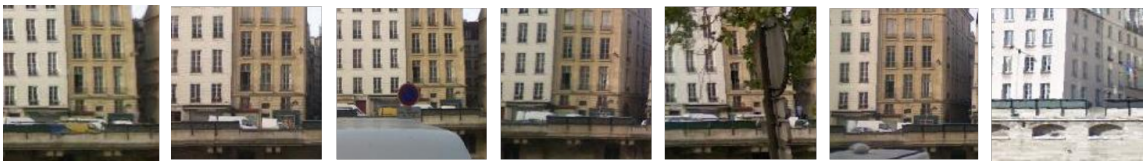
| FIM | Time (in min) |
|---|---|
| Database Generation (offline) | 32.58 |
| Mining | - |
| **Total (in mins)** | **0.0082** |

**Table 5.7** Time complexity when mining frequent patterns using Frequent Itemset Mining from a database of 200 images.

containing text. The type of object category discovered varies with the level at which we are mining. For example, “House Signs“ and “Street Lamps“ (Figure 5.7(b)) were discovered on the second level of mining, where the patch size is relatively low. As we move higher, the type of near-instance object categories discovered are those which cover a larger spatial extent, such as “Windows“ (Figure 5.7(c)) and “Text Banners“ (Figure 5.7(a)).

## 5.4 Summary

There are several object categories, which are near-instances. Objects in these categories exhibit less intra-class variance, which allows for instance retrieval techniques to be applied on them. We adapt the instance retrieval pipeline to solve tasks - category retrieval, and category mining, previous methods for which (SVM, FIM) are computationally expensive. In both cases, we are successfully able to retrieve/mine near-instance object categories. Our method, however, is restricted to near-instance categories alone. For other categories, considerable work needs to be done to bridge the gap between instance and category retrieval.



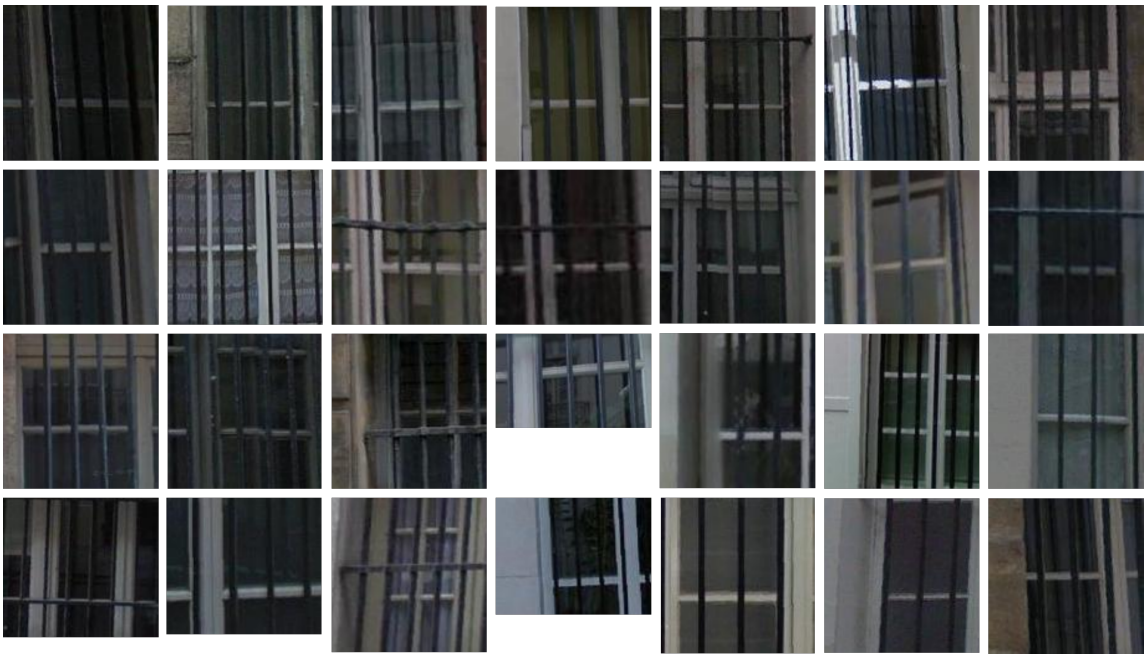
(a) Building Facades


(b) Multiple Windows


(c) Balcony with Text

(d) Street Lamps


(f) House Signs



(g) Grills

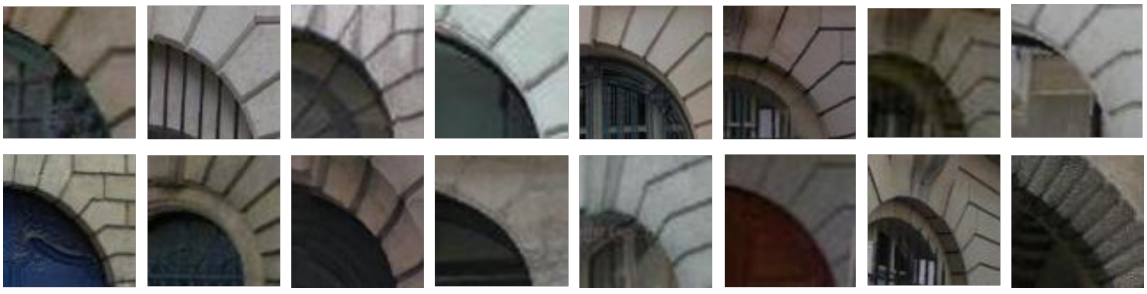
**Figure 5.8** Various near-instance Object Categories which were automatically discovered by our mining method

(a) Balconies

(b) Window Shutters

(c) Trees

(d) Pillars



(e) Semi-Ciruclar Arch

(f) Windows

**Figure 5.9** Various near-instance Object Categories which were automatically discovered by our mining method

*Chapter 6*

# Conclusions and Future Work

In this thesis, we attempt at solving various computer vision tasks by leveraging the existing mining algorithms. We study the previous attempts at employing traditional data mining schemes in computer vision, and the challenges associated with them.

**Prominent Person Mining in Photo Albums:** In Chapter 3 we describe an interesting application which, given a photo album off any social networking website, is able to accurately identify the most prominent person in the photo album, who is usually the person from whose profile the photographs have been obtained. Apart from being the most frequently occurring person, the most prominent person also exhibits several other characteristics - appearing in the center of many photographs, occupying a large area in photographs and often having several solo pictures of him/herself. All these characteristics are captured when identifying the most prominent person. Results have been demonstrated on synthetic as well as real world photo albums.

**Mining Characteristic Features for Architectural Style Categories** In Chapter 4 we delve into the relatively new area of architectural style categories. Recent efforts in image retrieval have been targeted towards retrieving instances of the same object or building. We move beyond the idea of buildings as instances, and explore the challenges associated with solving various computer vision tasks involving categories of architectural styles. We discover the utility of low-level feature configurations in retrieval and classification of monuments based on architectural styles,

**Discovering Semantic Patterns for Architectural Style Categories:** We further move towards automatically discovering high-level characteristics for categories of European architectural styles. We work with 5 prominent categories of European architecture. We discover structures such as Rose Windows, Pointed and Semi-Circular Arches. The association of the discovered structures with various styles of architecture has been verified from Wikipedia.

**Leveraging Instance Retrieval for Efficient Category Mining** In Chapter 5, we discuss the challenges inherent in the standard category retrieval/classification pipeline used today. On a separate front, much advances have been made in retrieving instances of the same object. Instances are retrieved in milliseconds. We show how several categories can be called as "near-instances" based on certain characteristics. We leverage the existing instance retrieval method to solve the problem of category re-

trieval/classification for a subset of object categories. We further propose a mining method to discover several near-instance object categories.

## 6.1 Future Work

There are several directions where one can go from here. The discovery of characteristic patterns from visual data is just the beginning. The importance of these patterns can further be utilized to solve various tasks in computer vision. For instance, mining the people who occur with the most prominent person in a photo album can enable us to build a social network which derives its basis from images itself. An auto-tagging application which detects the profile owner and automatically names him can be created.

We demonstrated the importance of contextual information when mining from visual data when mining the most prominent person. Various methods of semi-supervision can be explored, especially when working in Computer Vision. Images provide a lot of information, some of which is not obvious to the naked eye. Being able to correctly identify such information and its utility can have positive impacts over several mining solutions in Computer Vision - A bird has a higher probability of being found in the top half of the image, Objects in a particular scene have a specific layout with respect to each other, which can make discovery easier, if a person A is present in an image, there is a huge possibility that his close friend person B will also be found in the same image. These are just a few examples of how semi-supervision can benefit various computer vision problems.

There are several more ways in which the characteristic properties of the itemsets being mined can be defined. For example, it is not always the case that the most frequently occurring itemset is the most important one. In cases of fraud detection, burglary detection, diagnosis of a particular disease - all are situations in which the event occurs only rarely, however its implications are huge. Explored in the context of text mining under the banner of rare itemset mining, similar efforts can be made to identify such unusual occurrences from video surveillance feeds (burglary) or medical image/video data.

Mining from image and video data still faces the inherent challenges of location (both spatial and temporal) and scale. Rather than leveraging the existing text mining methods, we need to start from scratch and create custom-made algorithms which scale well to the huge sea of information which is stored in visual content. The emphasis is on both speed as well as accuracy.

As part of this thesis, we have made considerable effort towards the discovery of interesting patterns in a variety of data sets varying from photo albums downloaded from Facebook to images of European architecture obtained from Flickr.

# Related Publications

1. **Whose Album is This?** (*Oral*)
   Abhinav Goel and C.V. Jawahar.
   *In Proceedings of the Third National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics, India, 2011*

2. **Are Buildings only Instances? Explorations in Architectural Style Categories** (*Oral*)
   Abhinav Goel, Mayank Juneja and C.V. Jawahar.
   *Indian Conference on Vision, Graphics and Image Processing, 2012*

3. **Leveraging Instance Retrieval for Efficient Category Mining** (*Poster*)
   Abhinav Goel, Mayank Juneja and C.V. Jawahar.
   *Computer Vision and Pattern Recognition Workshops, 2013*

# Bibliography


[1]

[2] http://www.google.co.in/mobile/goggles/.

[3] http://lucene.apache.org/core/.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, 1994.

[5] R. Arandjelović and A. Zisserman. Smooth object retrieval using a bag of boundaries. In *ICCV*, 2011.

[6] R. Arandjelović and A. Zisserman. Smooth object retrieval using a bag of boundaries. In *IEEE International Conference on Computer Vision*, 2011.

[7] R. Arandjelović and A. Zisserman. Name that sculpture. In *ACM ICMR*, 2012.

[8] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[9] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[10] S. Belongie and J. Malik. Matching with shape contexts. In *IEEE Workshop on Content-based Access of Image and Video Libraries*, 2000.

[11] A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *CVPR*, 2005.

[12] D. M. Blei, A. Y. Ng, M. I. Jordan, and J. Lafferty. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:2003, 2003.

[13] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986.

[14] O. Chapelle. Training a support vector machine in the primal. *Neural Comput.*, 2007.

[15] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011.

[16] W.-T. Chu and M.-H. Tsai. Visual pattern discovery for architecture image classification and product image search. In *ACM ICMR*, 2012.

[17] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[18] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[19] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. A. Efros. What makes paris look like paris? *ACM Transactions on Graphics (SIGGRAPH)*, 31(4), 2012.

[20] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *TPAMI*, 2006.

[21] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE TPAMI*, 2010.

[22] F. M. Feng Tao and M. Farid. Weighted association rule mining using weighted support and significance framework. KDD, 2003.

[23] B. Fernando, l. Fromont, and T. Tuytelaars. Effective use of frequent itemset mining for image classification. In *ECCV*, 2012.

[24] 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*, 1981.

[25] A. Gilbert and R. Bowden. igroup: Weakly supervised image and video grouping. In *ICCV*, 2011.

[26] A. Gilbert, J. Illingworth, and R. Bowden. Action recognition using mined hierarchical compound features. *IEEE TPAMI*, 2011.

[27] A. Gilbert, J. Illingworth, R. Bowden, and G. X. England. Scale invariant action recognition using compound features mined from dense spatiotemporal corners. In *ECCV*, 2008.

[28] A. Gordo, J. A. Rodríguez-Serrano, F. Perronnin, and E. Valveny. Leveraging category-level labels for instance-level image retrieval. In *CVPR*, 2012.

[29] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOD*, 2000.

[30] T.-R. F. J. Hastie, Trevor. Hierarchical clustering, the elements of statistical learing (2nd ed.).

[31] D. Hauagge and N. Snavely. Image matching using local symmetry features. In *CVPR*, 2012.

[32] T. Hofmann. Probabilistic latent semantic indexing. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.

[33] M. Jain and C. V. Jawahar. Characteristic pattern discovery in videos. ICVGIP, 2010.

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

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

[36] S. R. Kiri Wagstaff, Claire Cardie and S. Schrodl. Constrained k-means clustering with background - knowledge. 2001.

[37] D. Liu, G. Hua, P. Viola, and T. Chen. Integrated feature selection and higher-order spatial feature extraction for object categorization. In *CVPR*, 2008.

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

[39] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal on Computer Vision*, 2004.

[40] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

[41] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image Vision Comput.*, 2004.

[42] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *ECCV*, 2002.

[43] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *ECCV*, 2002.

[44] N. Morioka and S. Satoh. Building compact local pairwise codebook with joint feature space clustering. In *ECCV*, 2010.

[45] N. Morioka and S. Satoh. Compact correlation coding for visual object categorization. In *ICCV*, 2011.

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

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

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

[49] T. Quack, V. Ferrari, B. Leibe, and L. J. V. Gool. Efficient mining of frequent and distinctive feature configurations. In *ICCV*, 2007.

[50] T. Quack, V. Ferrari, B. Leibe, and L. V. Gool. Efficient mining of frequent and distinctive feature configurations. In *ICCV*, 2007.

[51] T. Quack, V. Ferrari, and L. Van Gool. Video mining with frequent itemset configurations. CIVR, 2006.

[52] T. Quack, B. Leibe, and L. Van Gool. World-scale mining of objects and events from community photo collections. In *CIVR*, 2008.

[53] B. C. Russell, W. T. Freeman, A. A. Efros, J. Sivic, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR*, 2006.

[54] B. C. Russell, W. T. Freeman, A. A. Efros, J. Sivic, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. CVPR, 2006.

[55] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. *CoRR*, 2012.

[56] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman. Discovering objects and their location in images. In *ICCV*, 2005.

[57] J. Sivic, B. Russell, A. Zisserman, W. Freeman, and A. Efros. Unsupervised discovery of visual object class hierarchies. In *CVPR*, 2008.

[58] J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. In *CVPR*, 2003.

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

[60] J. Sivic and A. Zisserman. Video data mining using configurations of viewpoint invariant regions. In *CVPR*, 2004.

[61] S. Suzuki and K. Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 1985.

[62] M. Turk and A. Pentland. Face recognition using eigenfaces. In *CVPR*, 1991.

[63] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[64] P. Viola and M. J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 2004.

[65] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, 2005.

[66] B. Yao and L. Fei-Fei. Grouplet: A structured image representation for recognizing human and object interactions. In *CVPR*, 2010.

[67] L. Yu, J. Liu, and C. Xu. Descriptive local feature groups for image classification. In *ICIP*, 2011.

[68] M. J. Zaki. Scalable algorithms for association mining. *IEEE Trans. on Knowl. and Data Eng.*, 2000.

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

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