Interactive Visualization and Tuning of Multi-Dimensional Clusters for Indexing

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

Master of Science by Research
in
Computer Science

by

DASARI PAVAN KUMAR
200401021
d_pavan@research.iiit.ac.in

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

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled “Interactive visualization and tuning of multi-dimensional clusters for indexing ” by DASARI PAVAN KUMAR, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Prof. P. J. Narayanan

To my parents and brother

Acknowledgments

I am grateful to my advisor Prof. P.J. Narayanan for all the support he has provided over the past few years. His ideas and technical prowess always amazes me. He is one of the few persons who had a profound influence on my philosophy. He is always open to discussions, either personal or professional and gives enough freedom to take decisions. This I believe, has helped me a lot to evolve as a researcher and a student.

Murphy's law can truly be applied to my life as a masters student. I really thank my seniors Inam, Shanthan, Bama, Pawan Harish and Sesh who supported me a lot during those frustrating times. I bow to my spiritual guru who helped me realize what life is all about and allowed to mentally grow stronger.

I will always fondly remember my CVIT lab friends Sheetal, Naveen, Prachi, Dileep, Suhail, Chandrika, Wasif, Mrityunjay, Pratyush, Supreeth, Sreekanth, Rasagna, Chetan, Maneesh, Sandeep, Sai with whom I share so many wonderful memories. I would like to thank my lab seniors Shiben, Suryakant, Gopal, Vardhman, Nirnimesh and Ranta who were a great source of inspiration. I will remember all my juniors who used to have regular discussions.

Ofcourse, I thank my friends Avinesh, Kowshik, Rahul, Haritha, Pavan, Manohar, Praveen, Srivatsava, Ravi kiran, Kirthi, Shashank, Sarika, Ravi, Daka, Yaso, Samba, Prasant, Revanth, CVC, KP, Abhijit and Vamshi who made my stay at IIIT a very very memorable one. Last but not the least, I would be eternally grateful to my parents who provided me unconditional love and support since my day one on the Earth :).

Abstract

The automation of activities in all areas, including business, engineering, science, and government, produces an ever-increasing stream of data. Especially, the amount of multimedia content produced and made available on the Internet, both in professional and personal collections is growing rapidly. Equally increasing are the needs in terms of efficient and effective ways to manage it. And why is that so? Because, people believe that data collected contains valuable information. But, extracting any such information/patterns is however an extremely difficult task. This has led to a great amount of research into content based retrieval and visual recognition.

The most recent retrieval systems available extract low-level image features and conceptualize them into clusters. A conventional sequential scan on those image features would approximately take about a few hours to search in a set of hundreds of images. Hence, clustering and indexing forms the very crux of the solution. The state of the art uses the 128-dimensional SIFT as low level descriptors. Indexing even a moderate collection involves several millions of such vectors. The search performance depends on the quality of indexing and there is often a need to interactively tune the process for better accuracy. In this thesis, we propose a visualization-based framework and a tool which adheres to the it to tune the indexing process for images and videos. We use a feature selection approach to improve the clustering of SIFT vectors. Users can visualize the quality of clusters and interactively control the importance of individual or groups of feature dimensions easily. The results of the process can be visualized quickly and the process can be repeated. The user can use a filter or a wrapper model in our tool. We use input sampling, GPU-based processing, and visual tools to analyze correlations to provide interactivity. We present results of tuning the indexing for a few standard datasets. A few tuning iterations resulted in an improvement of over 5% in the final classification performance, which is significant.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The amount of digital data created worldwide is accelerating at an unprecedented, virtually incomprehensible rate. A study conducted by Hilbert et al. [33] documents the increase in global digital data between 1986 and 2007. They estimate that the current global storage capacity for digital information totals around 295 exabytes (an exabyte equals one billion gigabytes). The study suggests that computing storage capacity is growing at around 58 percent annually and the ability for enterprises to capture, collate and analyze organizational data is becoming simultaneously more important and difficult to manage as observed in Figure 1.5.

The automation of activities in all areas, including business, engineering, science, and government, produces an ever-increasing stream of data. The data is collected in very large databases because people believe that it contains valuable information. Extracting the valuable information, however, is a difficult task. When researchers have to analyze a new observational data set, they first try to learn what the data set looks like using descriptive modeling. Even with the most advanced data analysis systems, finding the right pieces of information in a very large database with millions of data items remains a difficult and time-consuming process. Hence, search is the only plausible way to find valuable information. But without an index, the search engine would scan every item in the corpus which would require considerable time and computing power.

A traditional sequential scan would approximately take about a few hours to search a set of tens of thousands of text documents. The additional storage required to store the index, as well as the increase in the time required for an update to take place, are traded off for the time saved during information retrieval. Search engine technology has had to scale dramatically to keep up with the growth of the web. One of the first web search engines WWWW (World Wide Web Worm) [57] had an index of 110,000 web accessible documents in 1994. As of 1997, the number of indexed documents rose to 100 million. By the end of 2003, Google [1] claimed to have indexed about 4 billion web documents. As of 2011, the indexed web contains atleast 14.6 billion pages [6]. In order to provide 'search results' in real time on such huge amounts of data, indexing is a must. Several datastructures have been used so far to meet the requirements of a particular search engine architecture. Some of them include suffix trees, inverted index, document-term matrix, Ngram, etc.

But with the advent of social media websites like Flickr, Youtube and Picasa, non-textual information like images and videos have seen an exponential growth over the last decade. In September 2010, Flickr reported to be hosting more than five billion images. Youtube approximately is growing with 20 hours of new video content per minute. These statistics are huge in comparision with the textual documents. With power comes responsibility. The power of sharing any video/image on the web gives rise to many concerns. Privacy of an individual can be easily compromised. Controversial and many a times objectionable content, like photos which contain nudity, videos which emotionally provoke a group of people can be uploaded anonymously. Copyrighted images could easily be reused. Its necessary to identify and censor images with skin-tones and shapes that could indicate the presence of nudity, with controversial results. Images/Videos are quickly becoming a wide spread medium for serving entertainment, education, communication, etc. Hence, there is an increasing demand to find suitable features to generate quick results for content based queries.

Lets roll back to data analysis. The process of finding right patterns cannot be fully automated since it involves human intelligence and creativity which are unmatchable by computers today. Humans will therefore continue to play an important role in searching and analyzing the data. Among other analysis methods for descriptive modeling, cluster analysis is most widely used to describe the entire data set by suggesting natural groups in the data set. Even though clustering algorithms produce useful clustering results, the cognitive understanding of the result is often not good enough to guide discovery since the result is statically represented in most cases, as is common in data mining applications. In dealing with such analysis, however, humans need to be adequately supported by the computer. One important way of supporting the human is visualisation of the data. Cognition of the clustering results can be amplified by dynamic queries and interactive visual representation methods, and understanding of the clustering results is transformed to another important data mining task - exploratory data analysis. Interactive information visualization techniques enable users to effectively explore clustering results and help them find the informative clusters that lead to insights.

Besides having a good descriptive model of multidimensional data sets, another challenging task is to identify important features or patterns hidden in the multidimensional space. We use the term, “feature” in a broader sense. What we mean by a “feature” is not only a dimension (or a variable) but also any interesting characteristics (e.g. clusters, gaps, outliers, and relationships between dimensions) of the data set. Dealing with multidimensionality has been challenging to researchers in many disciplines due to sparse nature of data and the difficulty in comprehending more than three dimensions to discover relationships, outliers, clusters, and gaps. This difficulty is so well recognized that its called “the curse of dimensionality”.

One of the commonly used methods to handle multidimensionality is the use of low-dimensional projections. Since human eyes and minds are effective in understanding one-dimensional (1D) histograms, two-dimensional (2D) scatterplots, and three-dimensional (3D) scatterplots, these representations are often used as a starting point. Users can begin by understanding the meaning of each dimension (since names can help dramatically, they should be readily accessible) and by examining the range and distri-

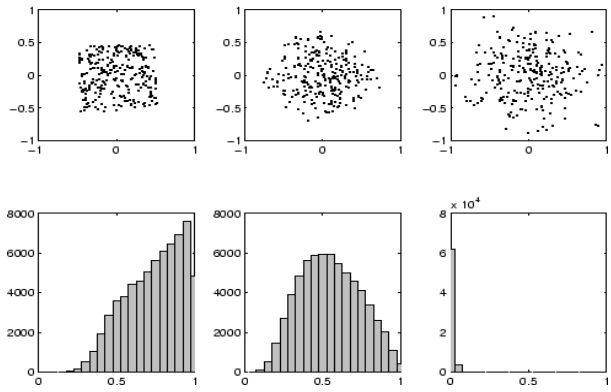

Figure 1.1 Curse of dimensionality illustrated with 256 d-dimensional points from a [0,1] uniform distribution with $d = 2$, 4 and 32 respectively. The top row shows the results of the 2D Principal Components Analysis (PCA). The bottom row displays how similarity (as a monotonically decreasing function of Euclidean distance) is distributed. As d increases, projections approach Gaussian distributions. An average pair of points' similarity decreases rapidly and similarities become approximately equal for most pairs with increasing d.

bution (normal, uniform, erratic, etc.) of values in a histogram. Then experienced analysts can suggest applying an orderly process to note exceptional features such as outliers, gaps, or clusters.

Next, users can explore two-dimensional relationships by studying 2D scatterplots and again use an orderly process to note exceptional features. Since computer displays are intrinsically two-dimensional, collections of 2D projections have been widely used as representations of the original multidimensional data. This is imperfect since some features may be hidden, but at least users can understand what they are seeing and come away with some insights.

Since the natural world is three dimensional, advocates of 3D scatterplots argue that users can readily grasp 3D representations. However, there is substantial empirical evidence that for multidimensional ordinal data (rather than 3D real objects such as chairs or skeletons), users struggle with occlusion and the cognitive burden of navigation as they try to find desired viewpoints. Higher dimensional displays have demonstrated attractive possibilities, but their display strategies are still difficult to grasp for most users.

The field of information visualization has found its utility in several areas like financial data analysis, digital libraries [53], manufacturing production control [15], crime mapping [81], etc. Constant efforts are made to create highly generic visualizations. But for domain specific problems, they often fail due to lack of sufficient focus. Treinish [75] proposes a task specific visualization design with application to weather forecasting. Stasko et al. [71] introduce a methodology for allowing designers to visualize or showcase the concurrency exhibited by parallel programs. Several other application specific visualization methods are proposed in the areas of medical imaging and computational fluid dynamics. However task specific those methods might be, they often tend to struggle if data explodes to millions of high dimensional vectors. We look to address this scenario by designing a framework which allows a user to generate high quality clusters in as less time as possible. We take a sub-problem of Content Based Image

Retrieval (CBIR) which clearly lacks necessary visual tools for interactive user feedback, to explain the framework. This led us to design and implement an interactive visualization tool.

There is an increasing interest in Content Based Image Retrieval over the past few years, since metadata-based systems are inherently limited. Textual information about images can be easily searched using existing technology, but requires humans to personally describe every image in the database. This is impractical for very large databases, or for images that are generated automatically. It is also possible to miss images that use different synonyms in their descriptions.

By now, we must have had a glimpse of what CBIR is all about. It is the process of retrieving desired images from a large collection based on image features and its uses can be found in

- Medical diagnosis
- Crime prevention
- Photograph archives
- The military
- Intellectual property
- Art collections
- Retail catalogs
- Architectural and engineering design
- Geographical information and remote sensing systems

Recent advancements in computer vision has introduced several image features to describe an image which can be roughly categorized into color, texture, local features and shape. In the context of matching and recognition, one needs a right combination of invariant region detectors and descriptors. Several region detectors like harris points, harris-laplace regions have been proposed. Various descriptors like SIFT [54], GLOH [59], shape-context [12], PCA SIFT [40] , spin images [50], steerable filters [25] have come into existence each with its own set of descriptions. Scale Invariant Feature Transform (SIFT) is one such local image features which made possible to achieve significant results in image retrieval and classification. These image features have many properties that make them suitable for matching differing images of an object or scene. The features are invariant to image scaling and rotation, and partially invariant to change in illumination and 3D camera viewpoint. They are localized pretty well in both the spatial and frequency domains, reducing the probability of disruption by occlusion, clutter, or noise. Large numbers of features can be extracted from typical images with efficient algorithms. In addition, the features are highly distinctive, which allows a single feature to be correctly matched with high probability against a large database of features, providing a basis for object and scene recognition. Following are the major stages of computation used to generate the set of image features.

- **Scale-space extrema detection:** The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation.
- **Keypoint localization:** At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.
- **Orientation assignment:** One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.
- **Keypoint descriptor:** The local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

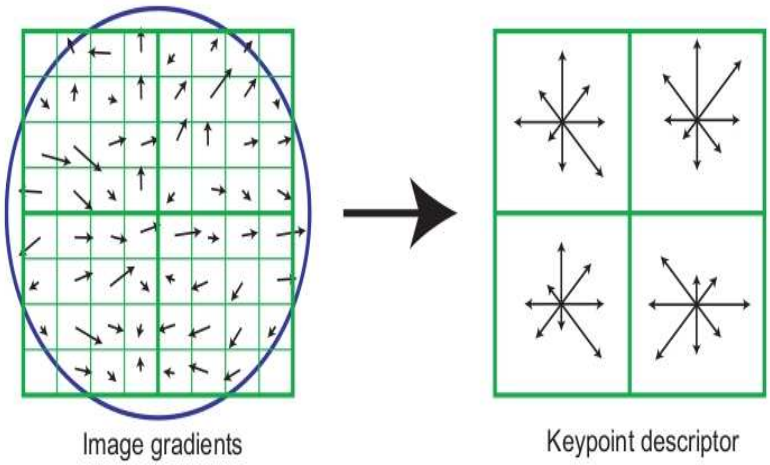

Figure 1.2 A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples, whereas current system computes 4x4 descriptors computed from a 16x16 sample array.

An important aspect of this approach is that it generates large numbers of features that densely cover the image over the full range of scales and locations. A typical image of size 500×500 pixels will give rise to about 1000 stable features (although this number depends on both image content and choices for various parameters). The quantity of features is extremely important for object recognition. The ability

to detect small objects in cluttered backgrounds requires that at least 3 features be correctly matched from each object for reliable identification. Hence, for a reasonably sized collection of a few thousands of images, the volume of data points explodes to a few million. For image matching and recognition, SIFT features are first extracted from a set of reference images either at interest points [55] or in a uniform grid [21] in a 128- dimensional space. and stored in a database. A new image is matched by individually comparing each feature from the new image to this previous database and finding candidate matching features based on Euclidean distance of their feature vectors.

Figure 1.3 A few Image classification categories

To maximize the performance of object recognition for small or highly occluded objects, we wish to identify objects with the fewest possible number of feature matches. This is where CBIR draws parallel with text search techniques. In document retrieval, each document is represented by a vector using a bag of words representation. A document vector is nothing but a histogram of words with each bin denoting a word's frequency in that document, disregarding grammar and word order. Analogous to this approach, for CBIR, each image is described by an image vector which is a histogram of *visual words*. Since each SIFT descriptor is a low-level feature, the entire set of descriptors extracted from the image collection is divided into a fixed number of clusters with each cluster center denoting a visual word (For detailed explanation, refer to [70]). SIFT descriptors in each cluster are similar in some sense of objectivity. To improve the retrieval accuracy using image vectors, Li et al. [52] incorporate learning methods like Support Vector Machine [74]

Significant results have been achieved so far on some datasets but do not fare well on practical image collections. Relevance feedback based systems were developed to over come such hurdles. However, retrieved results might not always convey the information required to boost learning parameters. In such a case, there is a need to fall back on information already available and to organize it better.

This motivated us in using CBIR as a sub-problem to design a framework for cluster analysis of large multidimensional datasets.

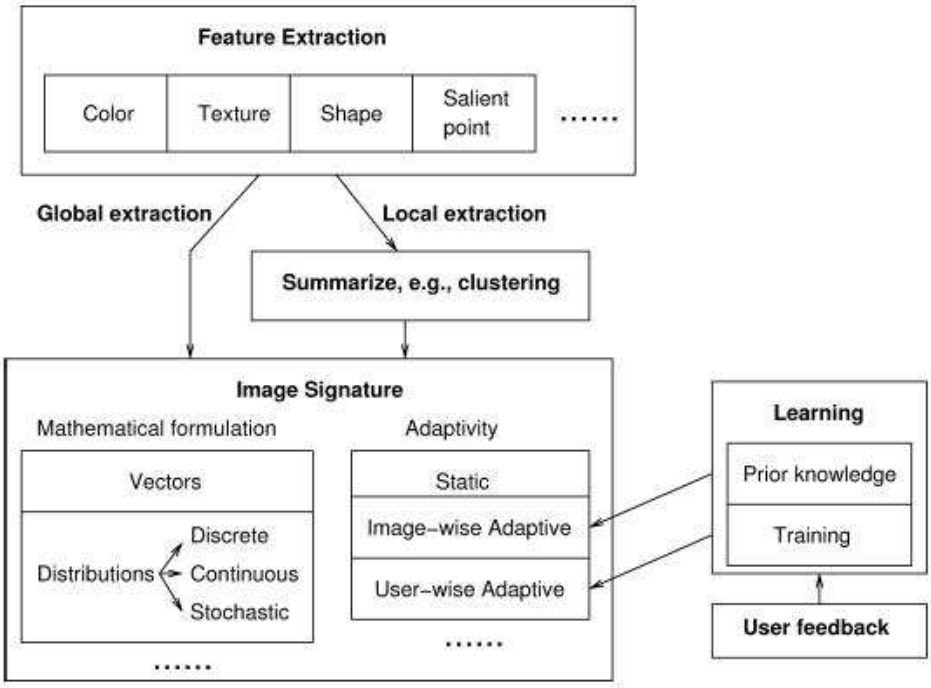

Figure 1.4 An Overview of CBIR process, by Datta et al. [23]

In this thesis, we redefine the CBIR process by incorporating a visual framework to generate user-favoured clusters of low-level image features like SIFT extracted from each of the images. We propose a visualization-based framework and a tool to help tune the clustering process with reference to indexing systems for CBIR. Generating qualitative clusters is possible if the user identifies the behaviour of underlying data points and controls the entire process interactively. We use a feature selection approach to improve the quality of clustering of SIFT vectors by weighing each dimension differently. Weights can be set interactively with automatic suggestions. We use a ranking schema and assign uniform weights to dimensional subset produced. The tool however supports both filter and wrapper model of clustering. For better exploration of unsupervised, multi-dimensional data, we provide one-dimensional projections, as well as two-dimensional projections where pair-wise relationships can be identified. We build on the idea of bin map and perform refinements to display the structure of SIFT descriptors. The tool also provides an interactive interface to analyze the clusters formed, by using a graph visualization scheme for the cluster centres as well as the vectors in each cluster. We use Euclidean Minimal Spanning Tree (EMST) proposed by Stuetzle [72]. This skeleton forms the basic layout representation of the data. Relative cluster validity techniques are implemented and visualized as a line chart to assist a user to understand the quality of clusters formed. We use a statistical sampling process in which a sampled subset of points from every image are used to tune the weights of each dimension semi-automatically.

To make the process interactive, a GPU is used for fast clustering as well as to compute the graph layout. We further provide the user an automatic weight suggestion scheme which proves handy in manual weight assignments. There are different ways of suggesting weights based on partition obtained from the clustering method and hence is used only as a supportive process. We observe a classification accuracy of 57.6% overall using our tool for UIUC dataset [2] consisting 15 different categories. Our main contribution is the combination of interactive visualization techniques to improve cluster quality for indexing problems. Thus, though the tool is specifically tuned for SIFT based indexing, it can be used for many learning-based problems that use high dimensional vectors.

An outline of our contributions in this thesis are as follows

- Proposed a visualization based framework for improving the process of clustering which forms the base of indexing huge datasets.
- Developed an interactive tool adhering to this framework and achieved better results in comparision to automatic methods.

This thesis has the following structure

- Chapter 2 provides the background required for our work. It widely starts with multi-dimensional visualization techniques like parallel coordinate plots and types of dimensionality reduction. We then focus mainly on cluster comparision schemas and graph drawing. We explain CUDA since it immensely helps us to speed up the process so that the tool is interactive.
- Chapter 3 describes the framework which is designed for cluster analysis. We discuss the fundamental one-dimensional and two-dimensional analysis methods for ranking feature space. We later provide interactive means for cluster analysis and automatic weight recommendation schemas to guide the process.
- Chapter 4 explains the environment in which a particular set of experiments are conducted and provide some insights which were previously not known. We numerically show that this framework has indeed resulted in better cluster quality there by leading to improved classification accuracy.
- Chapter 5 closes with insights and future work.

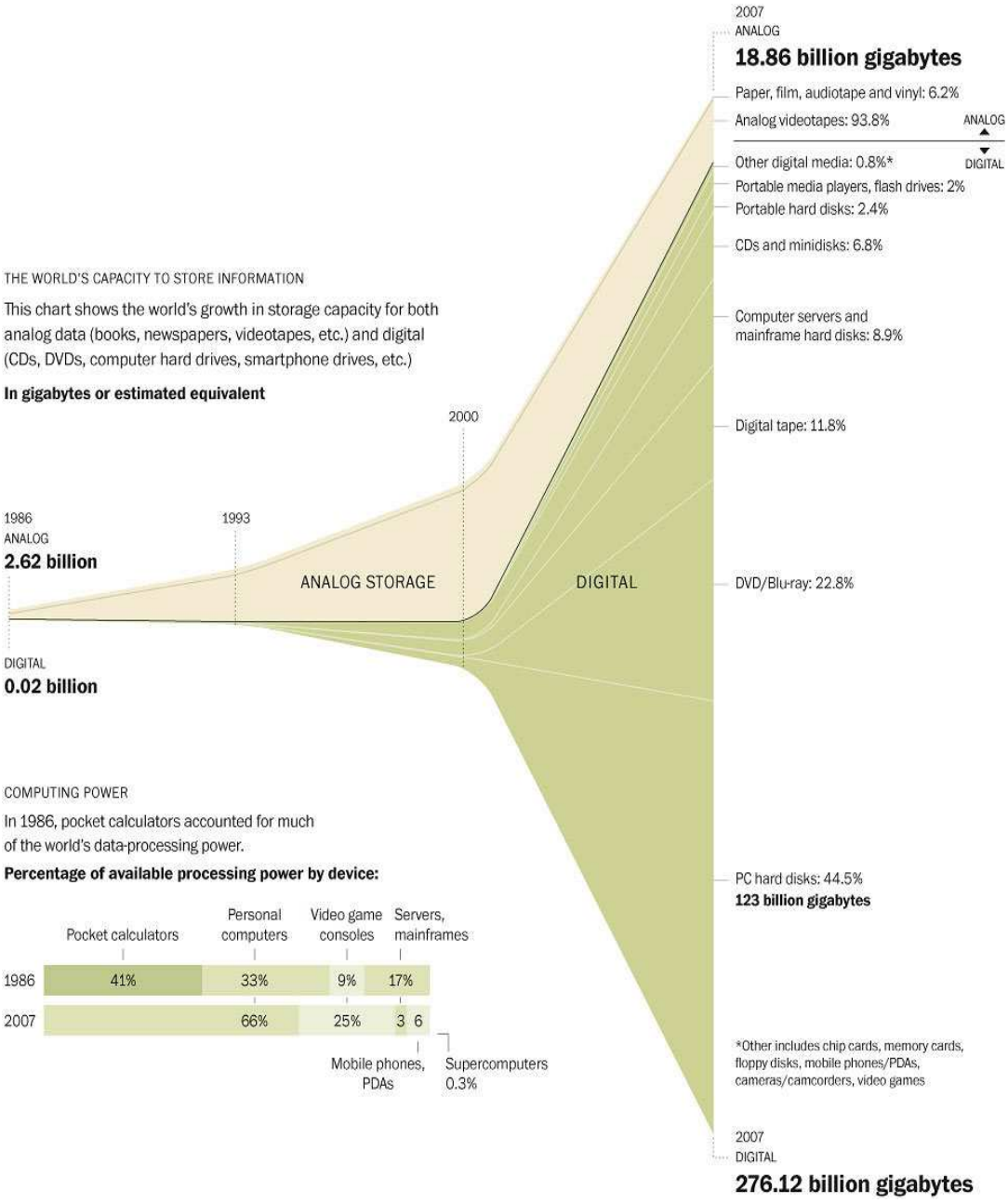

Figure 1.5 World's capacity to store information. (image courtesy: Washington Post.).

Chapter 2

Background and Related Work

We have so far discussed the rise of search engines and visual information, and the need to address the issue of content based image queries for several real world applications. A lot of research has gone into the problem of image querying using mathematical models and in many cases, using user feedback. Never has there been any attempt on incorporating visualization techniques to manually affect the underlying models generated, using human observational skills. But visualizing data is a task in itself and there is a vast amount of literature available. In our work, we attempt to generate better clusters of high dimensional SIFT vectors computed from sampled image set. In this chapter, we will take a look into the work that has previously gone into computing local image features, multi dimensional visualization techniques, dimensionality reduction, graph drawing, cluster comparisions and the more recent, Compute Unified Device Architecture (CUDA).

2.1 Image Representation

Many techniques have been used to represent the content of an image. All image classification and content based retrieval systems require an appropriate representation of the input images. One can represent an image globally or locally. Global appearance representation have problems with partial occlusions and background clutter. Local appearance representations, on the other hand, are at the heart of many highly efficient object recognition systems. Local features are computed around interest points or on a regular grid. Regular grid (dense representation) allows features to be computed at each sampled region on a dense grid. Dense representations can be used over sparse ones because regions with uniform texture, which usually are not returned by interest point detectors, will be represented equally well. But there is no general rule stating clearly the advantages of dense versus sparse representations. We choose to use sparse representation to reduce the amount of feature data being generated. This method requires that right image patches are chosen and described. It is in general carried out in two steps

- detecting interest points
- extracting feature descriptor from each interest region

A detailed comparision and overview of well known interest point detectors can be found in [58]. In our work, we use the Difference of Gaussians (DoG).

2.1.1 Difference-of-Gaussians

This involves convolving the image with a Gaussian at several scales, creating a so called scale space pyramid of convolved images. Interest points are now detected by selecting points in the image, which are stable across scales. For Difference-of-Gaussians (DoG) approach, the convolved images at subsequent scales are subtracted from each other. The DoG approach is in fact simply an approximation of the Laplacian. Stable points are searched in these DoG images by determining local maxima, which appear at the same pixel across scales.

In this work, the initial step is to compute SIFT feature descriptors at each of the interest points in an image. However, that is not the final step. We look to combine the entire set of such feature vectors to form a bag of words model.

2.1.2 Bag of Words model

In the last few years, bag of visual words have been commonly used in object recognition, object or texture classification, scene classification, image retrieval and related tasks. It directly relates to the bag of words model (BOW) originally used in text retrieval [9]. It has been introduced into the computer vision community by Sivic and Zisserman [70], who apply it to object retrieval in videos.

The BOW model is usually based on interest points and corresponding feature descriptions. It uses a clustering/vector-quantisation method to quantize the feature descriptors. Eventually each interest point is represented by an ID indexing into a visual-codebook or visual-vocabulary. Visual vocabularies are typically obtained by clustering the feature descriptors in high dimensional vector space. The dataset (or a subset of dataset) is clustered into k representative clusters, where each cluster stands for a visual word. The resulting clusters can be more or less compact, thus representing the variability of similarity for individual feature matches. The value of k depends on the application, ranging from a few hundred or thousand entities for object class recognition applications up to one million for retrieval of specific objects from large databases. For clustering, most often k-Means is used, but other methods are also used. Size of vocabulary is chosen according to how much variability is desired in the individual visual words. In object class recognition, the individual instances of a class can have large variations, while in retrieval for specific objects very similar features have to be found.

After vocabulary building, an image is then modelled as a bag of those so called visual-words. It can thus be described by a vector (or histogram) that stores the distribution of all assigned codebook IDs or visual words. Note that this discards the spatial distribution of the image features. The complete process for encoding an image with a visual vocabulary is summarized in Figure 2.1

We can observe that SIFT vectors are very high dimensional and not all dimensions can be equally interesting (curse of dimensionality). Hence we look to identify distributions in one or two dimensional

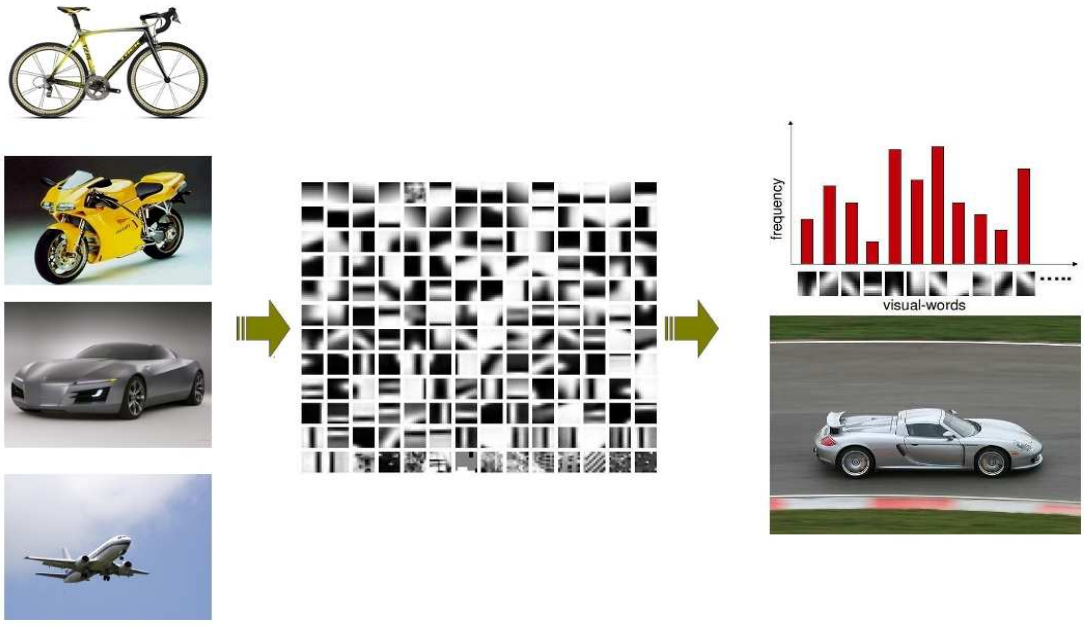

Figure 2.1 Bag of Words: Features are extracted from image dataset and clustered to get visual vocabulary/visual words collection. Using this vocabulary, each image can be represented as a set of frequencies of visual words.

space to take a more informed decision on choosing dimensions over which clustering can be performed. This takes us to the literature behind multi-dimensional visualization methods.

2.2 Multi-Dimensional Visualization Techniques

Several techniques exist for displaying multi-dimensional data. These include Scatterplot Matrices, Parallel Coordinate Plots, Pixel-Oriented displays, and Glyphs. We give an overview of each of these methods below.

2.2.1 Scatterplot Matrices

Scatterplots are simple plots used to compare two dimensional data by plotting points on a xy-Cartesian plane. Three dimensional data can be compared using a three dimensional scatterplot which would use the xyz-Cartesian space instead. For data that has more than three dimensions, these plots must be expanded to a matrix. A scatterplot matrix is a $N \times N$ matrix that has all the rows and columns labeled by the N dimensions (see Figure 4.4). Each cell (i, j) in the matrix is a scatterplot with the ith dimension on the y-axis and the jth dimension on the x-axis. Because this matrix has all N dimensions on the rows and columns, it is symmetric across the diagonal. This means that the cell (j, i) is the

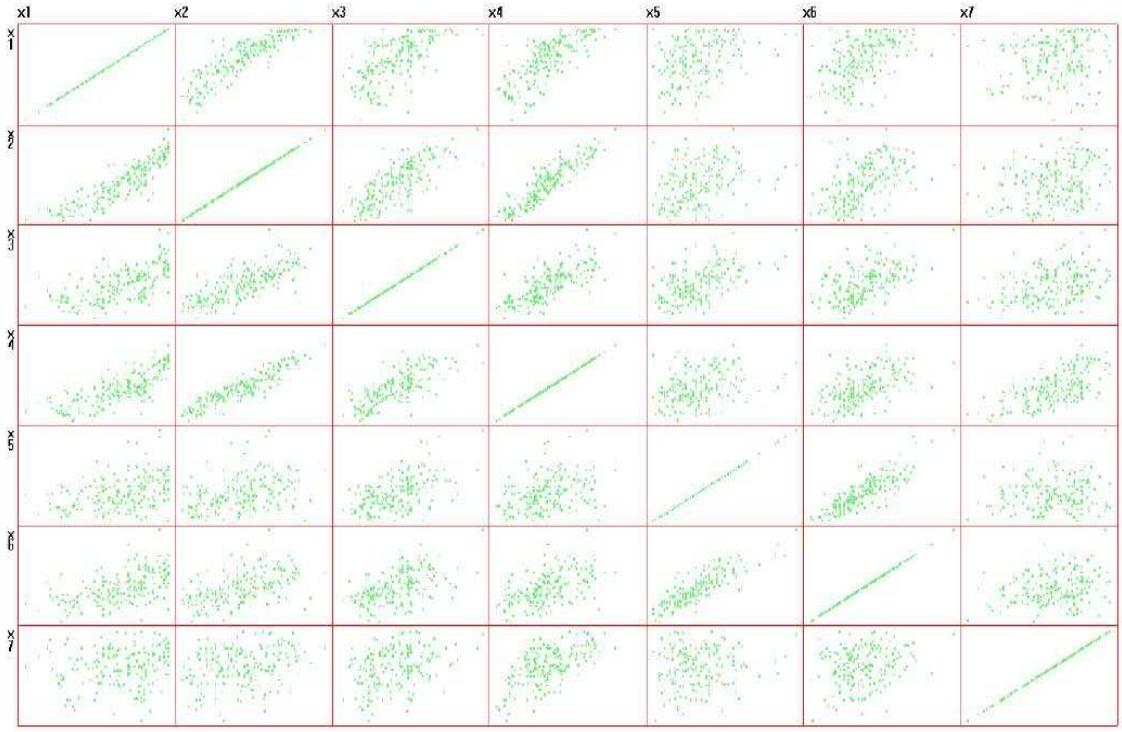

Figure 2.2 An example scatterplot matrix comparing variables corresponding to car models from XmdvTool [80]

same scatterplot as (i, j) except which of the two axes the dimensions are on. This technique was first presented in [36].

Scatterplot matrices work well for comparing a large number of records and dimensions. However, these matrices only provides information about how two dimensions relate. Comparing three dimensions requires an understanding about how each of the three dimensions relate to the other two dimensions.

2.2.2 Parallel Coordinate Plots

For scientists and others studying multi-variate relations or datasets, understanding the underlying geometry can provide crucial insights into what is possible and what is not. This need to augment our perception, limited as it is by the experience of our three-dimensional habitation, has attracted considerable attention into developing new visualization methods. Parallel Coordinate Plots is one such technique which supports to analyze the geometry of the data provided. Parallel coordinates creates a new coordinate system to represent n-dimensional objects [35]. This is done by placing each of the n-dimensions parallel to the y-axis (or the x-axis for horizontal axes) and evenly spacing them along the x-axis as shown in Figure 2.3 (or the y-axis if using horizontal axes). This creates a new coordinate system in the xy-plane that has n axes perpendicular to the x-axis. Each axis x_i is a dimension in the n-dimensional space. In other words, A vector V with values $(V_1, V_2, ...V_n)$ is visualized as a

polyline connecting points $(U_1, U_2..., U_n)$ on n vertical axes. A number of extensions to PCPs exist,

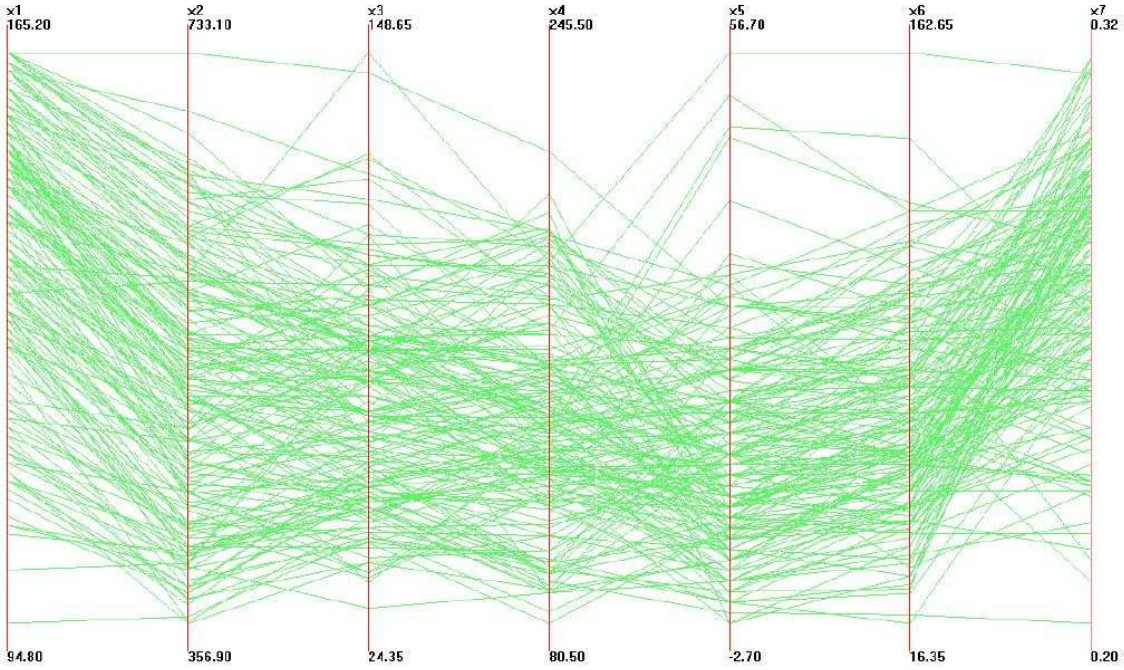

Figure 2.3 A sample Parallel Coordinate Plot analyzing correlations between various stock-market variables using XmdvTool [80]

like multiresolution and hierarchical [28] methods, 3D PCPs [38] and combinations of clustering, binning [8] and other features like outlier detection [39] to reduce visual clutter there by supporting large datasets.

2.2.3 Pixel-Oriented

Pixel-Oriented visualizations represent each record in the data set by a pixel. Due to the limited rendering area, this type of visualization has a limit on the size of the data set being visualized. Because of this limitation, these visualizations attempt to use the maximum number of pixels while still presenting an understandable picture. The techniques may be divided into query independent techniques that directly visualize the data (or a certain portion of it) and query dependent methods that visualize the data in the context of a specific query. Examples for the class of query independent techniques are screen filling curve and recursive pattern methods. The screen filling methods are based on Morton and Peano-Hilbert curve algorithms [20] and recursive patterns [41] based on generic recursive scheme, which generalises a wide range of pixel-oriented arrangements for visualizing large datasets. A sample pixel oriented display using recursive patterns is shown in Figure 2.4.

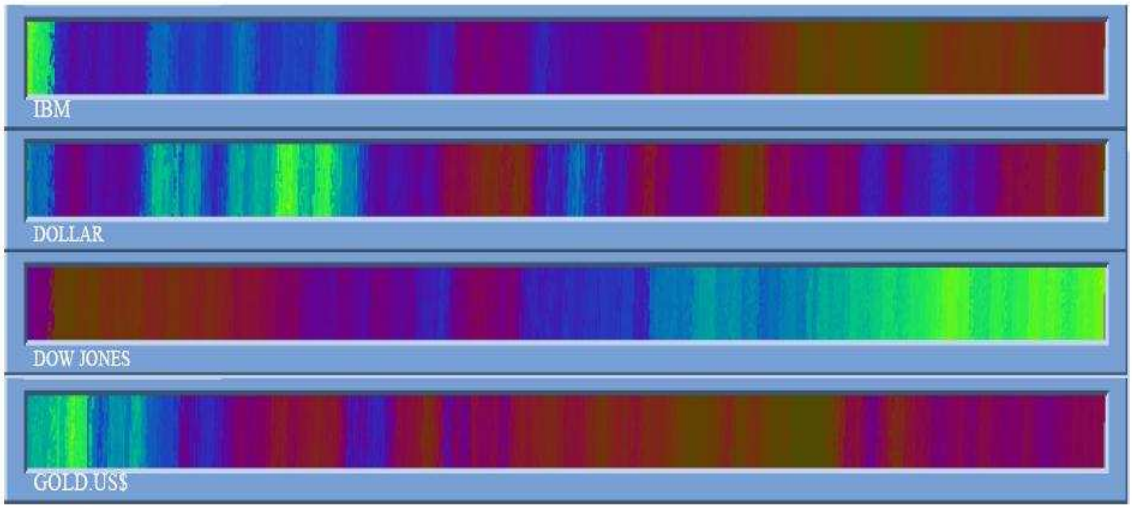

Figure 2.4 A sample recursive pattern based pixel oriented display showing horizontal arrangement (from [42])

2.2.4 Glyphs

Glyphs are icons that represent one record of the data set. Each dimension is mapped to one feature and the value determines some aspect of the feature. Two commonly used glyphs are Chernoff Faces [19] and Star Glyphs [51] (Figure 2.5). Like Pixel-Oriented visualizations, glyphs suffer from a lack of space to display all the records.

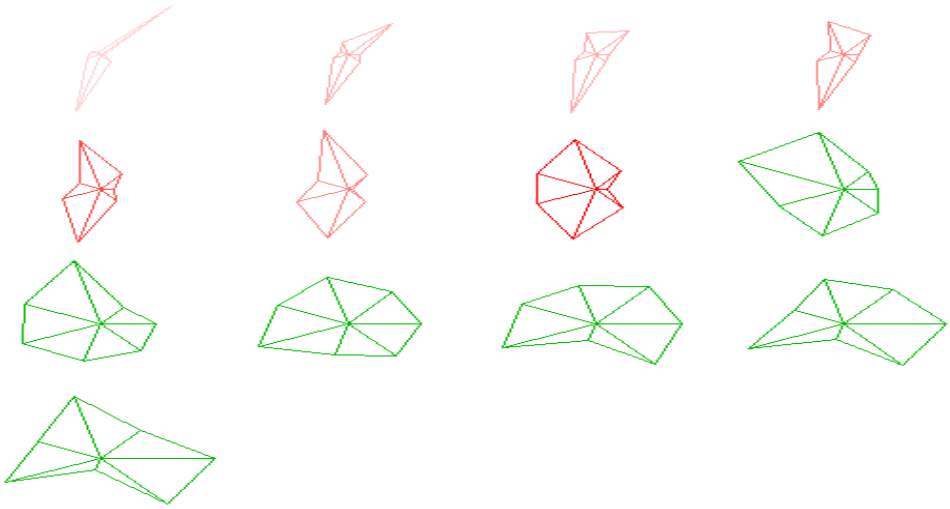

Figure 2.5 Star glyphs for Iris data points available with Xmdv tool [80]

Chernoff Faces use the human's ability to recognize small differences in faces to create a powerful visualization. Each dimension is mapped to a part of the face, such as the nose or the eyes, and the

shape or size is determined by the value of that dimension. Records that are similar would look the same, thus allowing for a simple and quick way to recognize clusters. It is also argued that analysis is done in parallel, which facilitates the efficient recognition of patterns [51].

Star Glyphs are based on whisker plots which have a central point and n lines, or whiskers, eminating from the central point. Each line represents one dimension of the data set whose value determines the length. The difference between whisker plots and star glyphs is that the ends of adjacent lines are connected to each other [51]. Recognizing clusters is simple with star glyphs since they will all have similar shapes.

2.3 Dimensionality Reduction

Techniques for clustering high dimensional data have used both feature transformation and feature selection methods. Feature transformation techniques attempt to summarize a dataset in fewer dimensions by creating combinations of the original attributes. These techniques are very successful in uncovering latent structures in datasets. However, since they preserve the relative distances between objects, they are less effective when there are large numbers of irrelevant attributes that hide the clusters in a sea of noise. Also, the new features are combinations of the originals and may be difficult to interpret in the context of the domain. On the other hand, feature selection methods select only the most relevant of the dimensions from a dataset to reveal groups of objects that are similar on only a subset of their attributes.

2.3.1 Feature Transformation

Feature transformations are commonly used on high dimensional datasets. These methods include techniques such as principal component analysis (PCA) (Figure 2.7) and singular value decomposition (SVD). The transformations generally preserve the original, relative distances between objects. In this way, they summarize the dataset by creating linear combinations of the attributes, and hopefully, uncover latent structure. Feature transformation is often a preprocessing step, allowing the clustering algorithm to use just a few of the newly created features. Some clustering methods have incorporated the use of such transformations to identify important features and to iteratively improve the clustering [76]. While often very useful, these techniques do not actually remove any of the original attributes from consideration. Thus, information from irrelevant dimensions is preserved, making these techniques ineffective at revealing clusters when there are large numbers of irrelevant attributes that mask the clusters. Another disadvantage of using combinations of attributes is that they are difficult to interpret, often making the clustering results less useful. Because of this, feature transformations are best suited to datasets where most of the dimensions are relevant to the clustering task, but many are highly correlated or redundant.

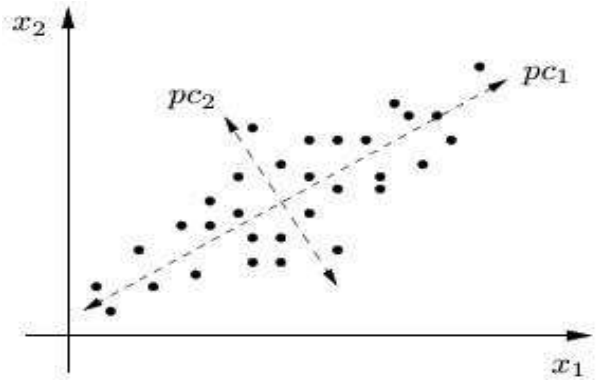

Figure 2.6 An example for Principal Component Analysis in two dimensional space

2.3.2 Feature Selection

Feature selection attempts to discover the attributes of a dataset that are most relevant to the data mining task at hand. It is a commonly used and powerful technique for reducing the dimensionality of a problem to more manageable levels. Feature selection involves searching through various feature subsets and evaluating each of these subsets using appropriate criterion [22] [69] [82]. The most popular search strategies are greedy sequential searches through the feature space, either forward or backward. The evaluation criteria follow one of two basic models: *filters* and *wrappers* [13].

The filter approaches evaluate the relevance of each feature (subset) using the data set alone, regardless of the subsequent learning algorithm. RELIEF [45] and its enhancement [48] are representatives of this class, where the basic idea is to assign feature weights based on the consistency of the feature value in the k nearest neighbors of every data point. Information theoretic methods are also used to evaluate features: the mutual information between a relevant feature and the class labels should be high [11]. Nonparametric methods can be used to compute mutual information involving continuous features [49].

On the other hand, wrapper approaches [46] invoke the learning algorithm to evaluate the quality of each feature (subset). Specifically, a learning algorithm (e.g., a nearest neighbor classifier, a decision tree, a naive Bayes method) is run on a feature subset and the feature subset is assessed by some estimate of the classification accuracy. Wrappers are usually more computationally demanding, but they can be superior in accuracy when compared to filters, which ignore the properties of the learning task at hand.

Both approaches, filters and wrappers, usually involve combinatorial searches through the space of possible feature subsets; for this task, different types of heuristics, such as floating search, beam search, bidirectional search, and genetic search have been suggested [17] [46], [65], [82] . It is also possible to construct a set of weak (in the boosting sense [26]) classifiers, with each one using only one feature, and then apply boosting, which effectively performs feature selection [78]. It has also been proposed to approach feature selection using rough set theory [47].

All of the approaches mentioned above are concerned with feature selection in the presence of class labels. Comparatively, not much work has been done for feature selection in unsupervised learning. Of course, any method conceived for supervised learning that does not use the class labels could be used for unsupervised learning; it is the case for methods that measure feature similarity to detect redundant features, using, for example, mutual information [68] or a maximum information compression index [60]. Different feature subsets and numbers of clusters, for multinomial model-based clustering, are evaluated using marginal likelihood and cross-validated likelihood in [77]. The algorithm described in [67] uses automatic relevance determination priors to select features when there are two clusters. In [22], the clustering tendency of each feature is assessed by an entropy index. A genetic algorithm is used in [44] for feature selection in k-means clustering. In [73], feature selection for symbolic data is addressed by assuming that irrelevant features are uncorrelated with the relevant features. Devaney et al. [24] describe the notion of "category utility" for feature selection in a conceptual clustering task. The CLIQUE algorithm [7] is popular in the data mining community and it finds hyper-rectangular shaped clusters using a subset of attributes for a large database.

The methods referred above "hard" feature selection (a feature is either selected or not). There are also algorithms that assign weights to different features to indicate their significance. For example, the method described by Pena et al. [63] can be classified as learning feature weights for conditional Gaussian networks. An EM algorithm based on Bayesian shrinking is proposed by Carbonetto et al. [16] for unsupervised learning.

2.4 Graph Drawing

Graph drawing or Graph layout, as a branch of graph theory, applies topology and geometry to derive two-dimensional representations of graphs. A drawing of a graph is basically a pictorial representation of an embedding of the graph in the plane, usually aimed at a convenient visualization of certain properties of the graph in question or of the object modeled by the graph. Very different layouts can correspond to the same graph. In the abstract, all that matters is which vertices are connected to which others by how many edges. In the concrete, however, the arrangement of these vertices and edges impacts understandability, usability, fabrication cost, and aesthetics. Different graph layout algorithms have been proposed so far which can be broadly classified into force based ([27] for example), spectral, orthogonal, symmetric, tree, hierarchical layouts. Early work on automatic graph layout and drawing is scattered through the computer science literature (for example [61], [66]). The first book devoted solely to graph drawing, by Battista et. al. [10], summarizes large areas of the field. The Graph Drawing conference series beginning in 1994 has resulted in proceedings that cover recent work in both systems and theory. The focus of part of this thesis is to provide a layout only, so we do not concentrate on the wealth of theoretical proofs about upper and lower algorithmic bounds: suffice it to say that most interesting computations on general graphs are NP-hard [14].

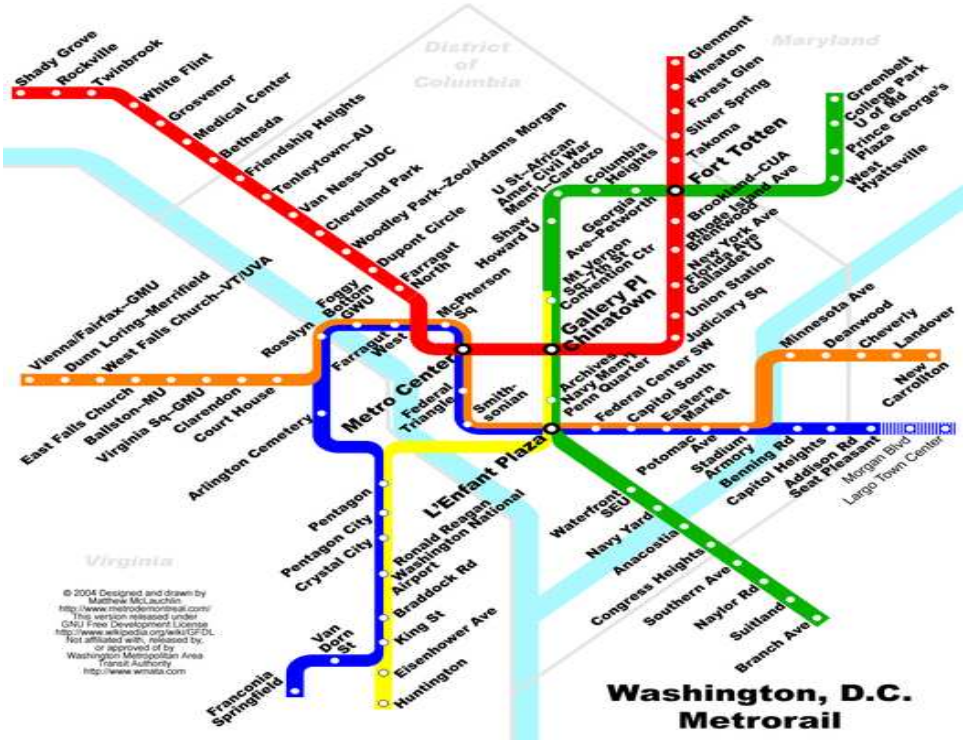

Figure 2.7 A graph layout of Washington D.C metro. (Image courtesy: Washington Metropolitan Area Transit Authority)

2.5 Evaluating Clustering Quality

The procedure of evaluating the results of a clustering algorithm is known under the term cluster validity. In general terms, there are three approaches to investigate cluster validity [31]. The first is based on external criteria. This implies that we evaluate the results of a clustering algorithm based on a pre-specified structure, which is imposed on a data set and reflects our intuition about the clustering structure of the data set. The second approach is based on internal criteria. In this case, the clustering results are evaluated in terms of quantities that involve the vectors of the data set themselves (e.g. proximity matrix). The third approach of clustering validity is based on relative criteria. Here the basic idea is the evaluation of a clustering structure by comparing it to other clustering schemes, resulting by the same algorithm but with different input parameter values. The two first approaches are based on statistical tests and their major drawback is their high computational cost. Moreover, the indices related to these approaches aim at measuring the degree to which a data set confixms an a-priori specified scheme. On the other hand, the third approach aims at finding the best clustering scheme that a clustering algorithm can define under certain assumptions and parameters. For more details on internal and external indices, refer to [31], [32].

2.5.1 Relative Indices

The fundamental idea of this approach is to choose the best clustering scheme of a set of defined schemes according to a pre-specified criterion. More specifically, the problem can be stated as follows.

Let P be the set of parameters associated with a specific clustering algorithm (e.g. the number of clusters n_c). Among the clustering schemes C_i, i=1,2,..n, defined by a specific algorithm, for different values of the parameters of P, choose the one that best fits the data set.

Then, we consider the following cases of the problem:

- P does not contain the number of clusters n_c, as a parameter.
- P contains n_c as a parameter.

In our current work, we do focus on crisp clustering, meaning that data point belongs only to a single cluster. So, we discuss validity indices suitable for crisp clustering.

2.5.1.1 The modified Hubert T-statistic

The definition of the modified Hubert T statistic is given by the equation

$$T = \frac{1}{M} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} Pr(i,j).Q(i,j)$$

where N is the number of objects in a dataset, $M = N(N-1)/2$. Pr is the proximity matrix of the dataset and Q is a N x N matrix whose (i,j) element is equal to the distance between the representative points (V_{ci}, V_{cj}) of the clusters where the objects X_i and X_j belong.

2.5.1.2 The Davies-Bouldin (DB) index

A similarity measure R_{ij} between the clusters C_i and C_j is defined based on a measure of dispersion $s_i$of a cluster C_i and a dissimilarity measure between two clusters d_{ij}. The R_{ij} index is defined to satisfy the following set of conditions:

- $R_{ij} \geq 0$
- $R_{ij} = R_{ji}$
- If $s_i = 0$ and $s_j = 0$ then $R_{ij} = 0$
- If $s_j > s_k$ and $d_{ij} = d_{ik}$ then $R_{ij} > R_{ik}$
- If $s_j = s_k$ and $d_{ij} < d_{ik}$ then $R_{ij} > R_{ik}$

These conditions state that R_{ij} is non-negative and symmetric. A simple choice for R_{ij} that satisfies the above conditions is:

$$R_{ij} = \frac{(s_i + s_j)}{d_{ij}}$$

Then the DB index is defined as

$$DB_{n_c} = \frac{1}{n_c} \sum_{i=1}^{n_c} R_i$$

$$R_i = \max_{i=1,2...n_c, i \neq j} R_{ij}$$

It is clear from the above definition that DB_{n_c} is the average similarity between each cluster c_i, (for i=1,2..n_c) and its most similar one. It is desirable for clusters to have the minimum possible similarity to each other; therefore we seek clusterings that minimize DB index. The DB_{nc} index exhibits no trends with respect to number of clusters and thus we seek the minimum value of DB_{nc} in its plot versus the number of clusters.

Other validity indices have been proposed in [32]. The implementation of most of these indices is computationally very expensive, especially when the number of clusters and objects in the data set grows very large. Indices like RMSSTD and RS can be computed to evaluate clusters. The idea here is to run the algorithm a number of times for different set of parameters and search for a "knee" in the corresponding graph plot.

$$RMSSTD = \frac{\sum_{i=1}^{n_c} \sum_{j=1}^{v} \sum_{k=1}^{n_{ij}} (x_k - \overline{x_j})^2}{\sum_{i=1}^{n_c} \sum_{j=1}^{v} (n_{ij} - 1)}$$

$$RS = \frac{SS_b}{SS_t} = \frac{SS_t - SS_w}{SS_t}$$

$$SS = \sum_{i=1}^{N} (X_i - \overline{X})^2$$

where SS means sum of squares, x_k is a data point ,n_{ij} is the number of data points of dimension j in cluster i, $\overline{x_j}$ is mean of data points in dimension j and

- SS_b refers to the sum of squares between groups
- SS_b refers to the sum of squares within group
- SS_t refers to the total sum of squares, of the whole dataset

2.6 GPU and Compute Unified Device Architecture

Graphics Processing Units have massively parallel processors set up as a parallel architecture. The graphics pipeline is well suited to rendering process as it allows a GPU to function as a stream processor.

Nvidia 8 series and later GPUs with CUDA programming model provide an adequate API for non-graphics applications. CUDA is a programming interface which tries to exploit the parallel architecture of GPUs for general purpose programming. It provides a set of library functions as extensions of C language. The CPU sees a CUDA device as a multicore co-processor. The design does not restricts the memory usage as was in the GPGPU case. This enhanced memory model allows programmer to better exploit the inherent parallel power of GPU for general purpose computations.

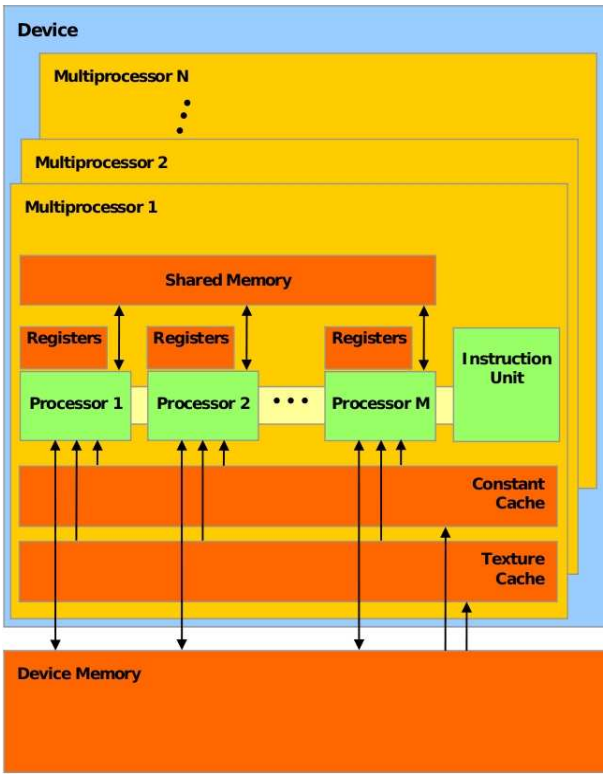

Figure 2.8 CUDA hardware model

At the hardware level, GPU is a collection of SIMD multiprocessors with several processors in each as shown in Figure 2.8. For example, Nvidia 8 series has eight processors in each multiprocessor. Each multiprocessor contains a data parallel cache or shared memory, which is shared by all of its processors. It has texture cache and read-only constant cache that is shared by all processors. A set of local 32 bit registers are available per processor. Each processor in a multiprocessor executes the same instruction in every cycle. The multiprocessors communicate through device or global memory. The processing elements of a multiprocessor can synchronize with one another, but there exists no direct synchronization mechanism between multiprocessors.

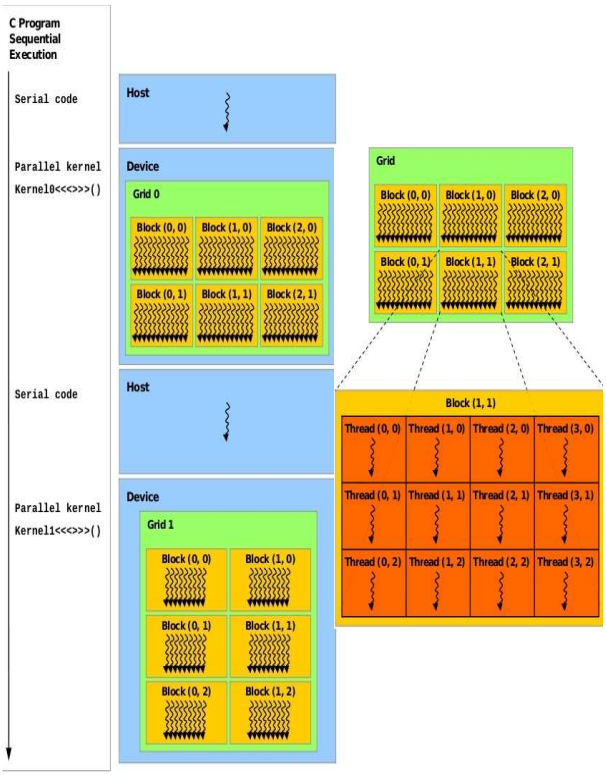

Figure 2.9 CUDA programming model

At the software level, for a programmer, CUDA model is a collection of threads running in parallel. A warp is a collection of threads that are scheduled for execution simultaneously on a multiprocessor. The warp size is fixed for a specific GPU. The programmer can select the number of threads to be executed. If the number of threads is more than the warp size, they are time-shared internally on the multiprocessor. A collection of threads run on a multiprocessor at a given time which is called a block. Multiple blocks can be assigned to a single multiprocessor for time shared execution. They also divide the common resources like registers and shared memory equally among them. A single execution on a device generates a number of blocks. The collection of all blocks in a single execution is called a grid. Each thread and block is given a unique ID that can be accessed within the thread during its execution. All threads of the grid execute a single program called the kernel.

Chapter 3

Framework and VisTool

Few attempts have been made to incorporate more user subjectivity into the summarization of low-level features. One of the key contributions of this thesis is the redefinition of CBIR process by incorporating a visual framework to generate user-favoured clusters of low-level features like SIFT extracted from each of the images (Figure 3.1). A learning based CBIR system relies on the quality of image vectors and relevance feedback provided by the user to train the classifier. The visual words should be of high quality. Generating qualitative clusters is possible if the user identifies the behaviour of underlying data points and controls the entire process interactively.

Visualization of large amounts of high dimensional data has been an active research area in information visualization. Keim [43] gives a good overview and categorization of relevant visualization techniques. Many of these methods aim at the identification of interesting formations in the data, such as linear correlations. However, to analyze SIFT descriptors extracted over a collection of images, these do not apply directly. Thus, we aim at presenting a visualization based framework with an implemented tool to tune the entire process of indexing SIFT.

Interactive analysis of such huge abstract datasets is not possible using individual traditional methods. We sample the entire dataset to a manageable size using a stratified sampling method. We follow a feature selection approach and incorporate a rank-by-feature schema to identify subspaces. Distribution along each of the dimensions can be analyzed by its corresponding histogram and box-plot. We use a Parallel Coordinate Plot widget to enable a user to identify two-dimensional correlations. Once dimensions are analyzed, suitable weights are chosen to generate visual words. As the framework suggests, a user is free to choose either a filter or a wrapper model of clustering which can be incorporated as a plugin in the tool. Whatever might be the method used to compute visual words, it is necessary to analyze the cluster structure formed. Graph layout provides an excellent means of analyzing such high dimensional structures. Since it is an unsupervised clustering process, users must be able to evaluate the resulting clusters formed using some qualitative measure. Relative index measures form an excellent choice. User can interactively re-assign weights to each feature vector based on his observation of a clustering process. Some automatic methods of suggesting weights have also been included to aid the choice. The procedural framework after we sample a dataset is described as follows.

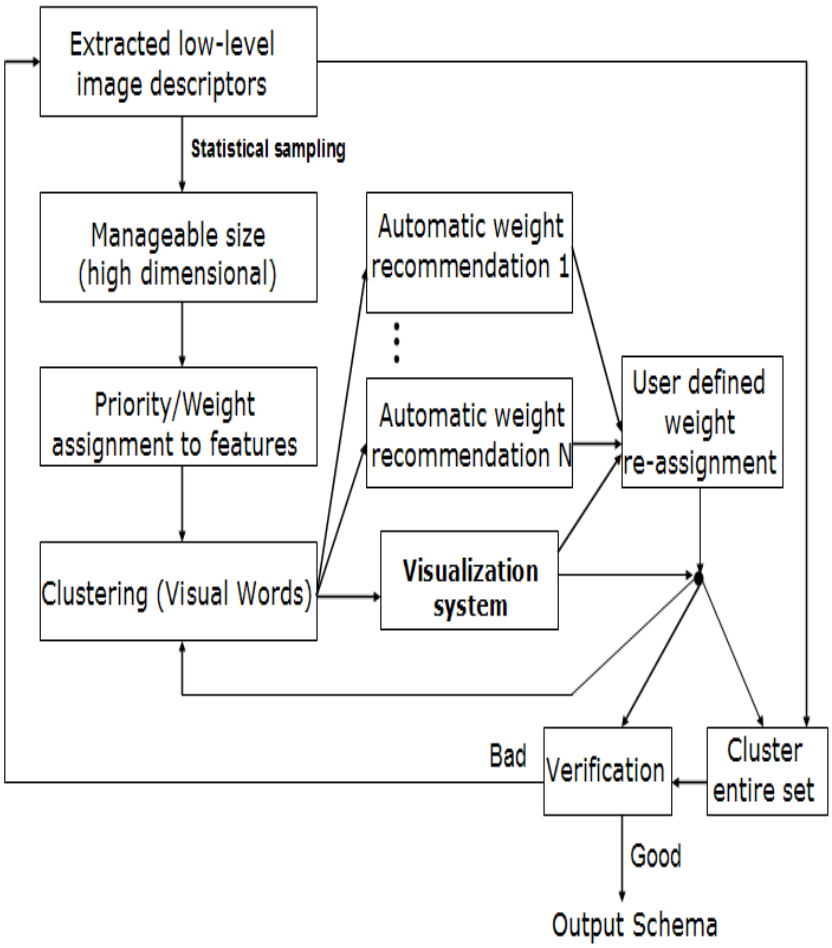

Figure 3.1 Framework: Overview of the procedure

3.1 Feature Selection and Weight Assignment

We integrate four ranking criteria into our tool, since they are common and fundamental measures for distribution analysis.

- Entropy of the distribution (0 to ∞)
- Normality of the distribution (0 to ∞)
- Number of potential outliers (0 to N)
- Number of unique values (0 to N)

We use an entropy measure to compute uniformity. Given k bins in a histogram H, its entropy E is defined as

$$E(H) = -\sum_{i=1}^{k} P_i \log_2(P_i) \tag{3.1}$$

where P_i is the probability that an item belongs to i-th bin. We chose *omnibus moments test* for normality from several statistical tests available. Several outlier detection algorithms have been proposed in the field of data mining [64]. We select an item of value d to be an outlier if, $d >$ (Q_3 + 1.5*IQR) or $d <$ (Q_1 - 1.5*IQR) where IQR is the interquartile range (defined as the difference between the first quartile (Q_1) and the third quartile (Q_3)).

We can however notice that any filter model based feature selection method can be incorporated instead of a ranking schema and assign uniform weights to the subset produced.

3.1.1 One-dimensional Distribution Analysis

Users may begin their exploratory analysis by scrutinizing each dimension one by one. A mere look into the distribution of values of a dimension gives useful insights. The tool provides *histograms* and *boxplots* for graphical display of 1D data as shown in Fig. 3.2. Histograms graphically reveal the skewness and scale of the data. Boxplots provide a quick way of examining one or more sets of data graphically by showing a five-number summary

the minimum: smallest observation of the sample

the first quartile: cuts off lowest 25% of data = 25th percentile

the median: cuts data set in half = 50th percentile

the third quartile: cuts off highest 25% of data, or lowest 75% = 75th percentile and

the maximum: highest observation of the sample

These numbers provide an informative summary of dimension's center and spread and thus help in selecting dimensions for deriving a model.

The sub-interface consists of four coordinated parts: the rank box, the table view, the order view and the histogram view. Users can select a ranking criterion from the rank box, which is a combo box, and then see the overview of scores for all dimensions in the table view (Figure 3.3) according to the ranking schema selected. All dimensions in the data table are sorted in decreasing value of scores on default. The table view consists of three columns. The first column denotes the dimension name, second denotes the score of that dimension according to ranking schema, and the third column in the table view displays weight assigned to it by user. (discussed in clustering framework).

The order view is a bar chart, where in the length of the bar denotes the rank of that dimension. All dimensions are aligned from top to bottom in the original order and each dimension is color coded by corresponding weight-value (third column in the table view). Weight is directly proportional to the color scale which is displayed below the order view. Its mapping can be obtained by a simple mouse hover over the display bar (Fig. 3.4). A mouse hover over a bar in the order view displays the corresponding dimension id in a tooltip window. Thus, user can preattentively identify dimensions of highest and lowest rank and observe the overall pattern. The user is provided an option to interchange the parameters in the order view such that color of each bar denotes the rank and length, its weight. The histogram view is a combination of one-dimensional histogram plot and box-plot to display 1D projections.

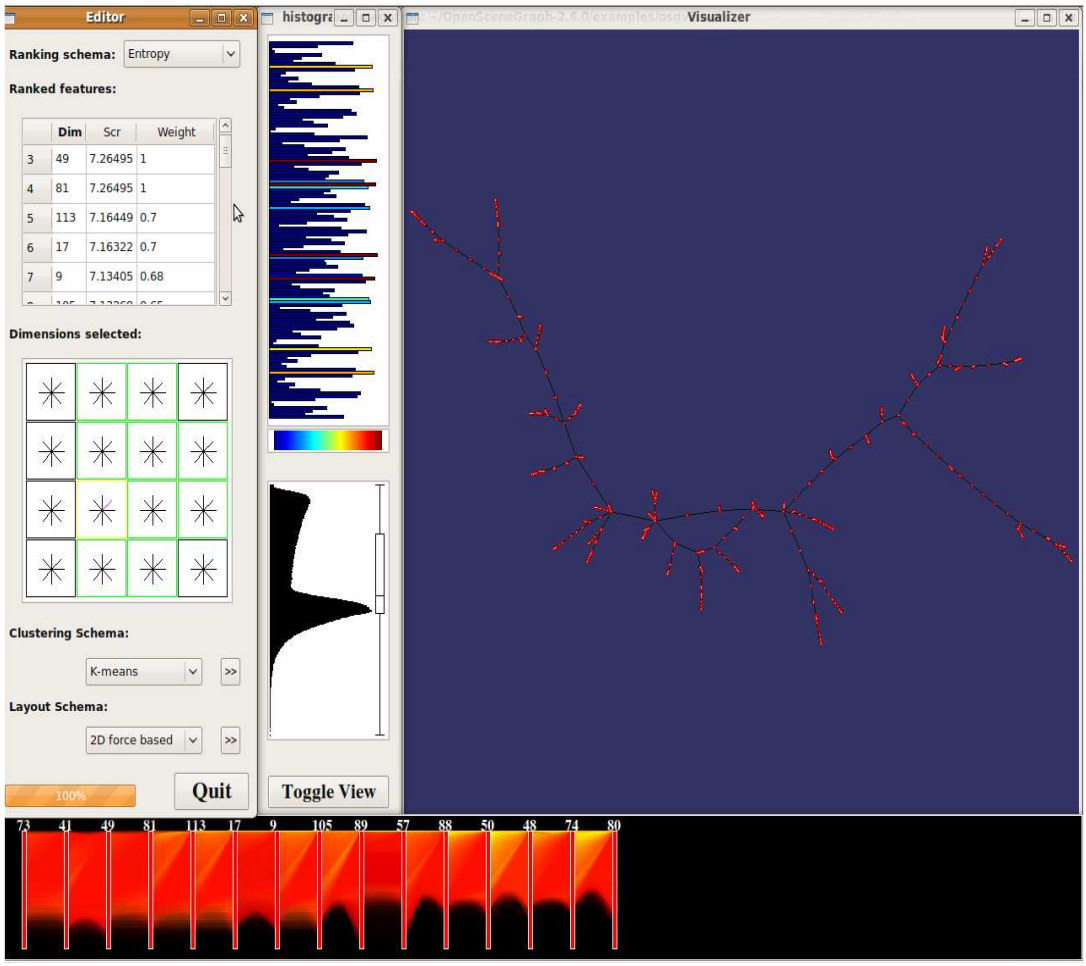

Figure 3.2 Sift-visualizer: tool adhering to the framework

The mouseclick event in the rank view or a cell activate event in the table view is instantaneously relayed to other views. The corresponding item is highlighted in the rank and the table view, and its histogram and box plot is rendered in the histogram view. In other words, a change of dimension in focus in one of the rank or table view leads to instantaneous change of dimension in focus in other component views.

3.1.2 Identifying Correlations

For better exploration of unsupervised, multi-dimensional data, after scrutinizing one-dimensional projections, it is natural to move on to two-dimensional projections where pair-wise relationships can be identified. Based on our discussion in section 2, we build on the idea of bin map and perform refinements to display the structure of SIFT descriptors.

Parallel Coordinate Plots have little to gain from high precision floating point representation of data. When data values are rounded to a lower precision representation, the maximum erroneous displacement

	Dim	Scr	Weight
3	49	7.26495	1
4	81	7.26495	1
5	113	7.16449	0.7
6	17	7.16322	0.7
7	9	7.13405	0.68

Figure 3.3 table view: Ranked dimensions are displayed in decreasing order of *score* computed by the ranking criterion.

Figure 3.4 Colormap: Left most color represents lowest value and right most, the highest value.

that a line in a plot will have is directly related to the rounding error. The axes of PCP currently are less than 127 pixels. Hence, a quantization to 8-bit values will yield a maximum displacement of a single pixel which doesn't show any effect in the current scenario. This step reduces the data from 4 bytes to a single byte per point per attribute, greatly reducing the necessary storage.

A PCP without any selections can be quickly generated solely from the joint-histograms of the data. We make use of binning approach proposed by Artero et. al. [8]. Only the joint histogram between each pair of neighbouring axes is needed to build the parallel coordinate plot using this technique. Fast exploration of data is made possible by computing joint histograms over all pairs of axes. For N axes, we get $\frac{N(N-1)}{2}$ histograms for all pairs. We adopt the rendering approach of Philipp et. al. [62] where histogram bins form a direct basis for drawing the primitives. Instead of having to draw a line for each data point, only a single primitive is drawn for each histogram bin. We use additive blending to combine all drawn primitives. We use a square-root intensity scale, as to prevent over-saturation of high-density areas in the plot, while keeping a good visual contrast in low intensity areas.

A value change event in the weight column in the table view is instantaneously relayed to PCP display. If a weight W ($>$0) is assigned to a dimension by the user, the corresponding joint histogram is rendered in order of precedence of selection. For example, from Figure 3.2, say dimension 113 is assigned a weight, then a joint histogram between 81 and 113 is rendered. Next, when dimension 17 is given some weight, a bi-histogram between 113 and 17 is loaded into memory and rendered in PCP display.

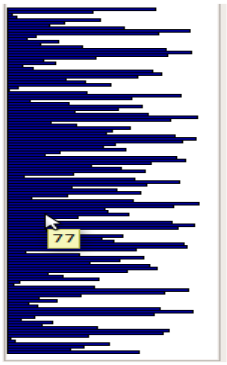

Figure 3.5 Rank View: The Length of a bar denotes its corresponding rank according to the score obtained from ranking schema. Color represents its current weight.

Thus, we provide a user with the power to analyze multivariate structure of *interesting* dimensions in 1D projections. User might find 1D projection of a dimension to be interesting, but it might not have any correlation with other dimensions of significant interest. In such a case, user can revert back to remaining dimensions by deselecting the uncorrelated dimension. De-selecting a dimension can be performed simply by reassigning its corresponding weight W to zero in the table view.

3.1.3 Weight Assignment Using Glyphs

Since we consider scale invariant feature transform descriptors, the number of dimensions present are 128. If we observe how SIFT is computed, we notice that each interest point (which is computed using some interest point detector like DoG [58]) is divided into a 4×4 matrix where cell size depends on the scale computed by interest point detector. Eight orientation angles are chosen describing the infomation in each cell. Following a similar approach, we display a 4×4 matrix of glyphs where each glyph is further divided into eight orientations as shown in Figure 3.2. Šaltenis et. al. use glyphs to display information about k-dimensional data [79].

A Mouseclick event in the glyph view generates a zoomed-in version of the corresponding cell in which mouseclick event occured. User can visually assign weights between neighbouring dimensions as shown in Figure 3.6. This mode of assigning weights is very useful if two neighbouring dimensions are of sufficent interest and user can visually approximate priorities. A left mouseclick event in the orientation view assigns a weight to each of the corresponding adjacent dimensions with respect to the angle selected. A right mouseclick deselects previously assigned weights in the cell. The updated weights are relayed to the table view on either of the events.

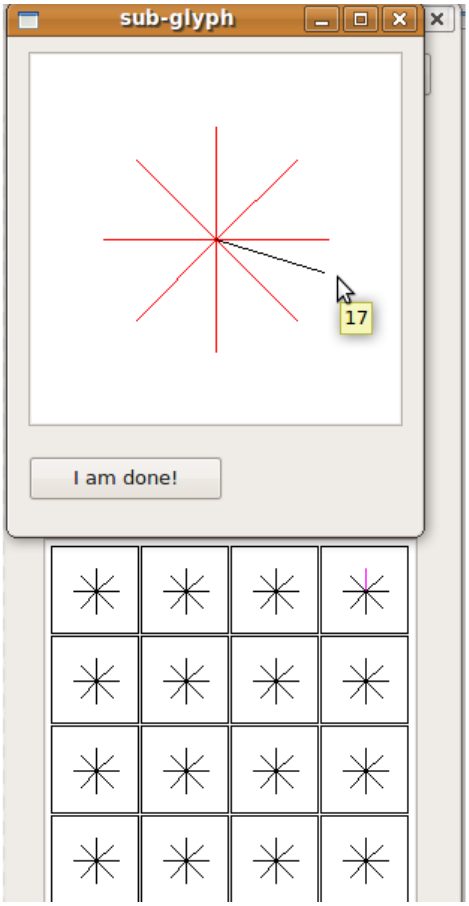

Figure 3.6 Orientation view: Visually assigning weights to adjacent dimensions. Greater the inclination, more the weight given to an orientation.

3.2 Data Clustering

After potentially identifying a weighted sub-space, user clusters the set of data points. Since different users might be interested in different clustering methods, it is desirable to allow users to customize the available set of clustering schemas. However, we have chosen the standard k-means [37] as a starting point and implemented it on the GPU using CUDA [3] to achieve significant speed up. A modified Euclidean distance schema is incorporated into computing distance between cluster means and points. Distance between a cluster mean P and a data point Q is computed using the formula

$$D_{(pq)} = \sqrt{\sum_{i=1}^{128} W_i * (P_i - Q_i)^2}, \text{ where } (0 \leq W_i \leq 1) \tag{3.2}$$

denotes weight assigned to dimension i in the table view. After every iteration, we update cluster means as in the standard procedure.

3.3 Visualization for Cluster Analysis

Clustering large amounts of data though reduces the information overhead, generates new points in feature space which need to be analyzed. Usual mathematical process helps in finding out the quality of points generated but cannot give an overview of how they are related to each other. Visualization of such points might help in providing better insights into the cluster quality. Hence we build an interactive graph layout interface for visual analysis.

3.3.1 Graph Layout

Often, clustering algorithms go hand in hand with graph drawing methods providing important means of dealing with increasingly large datasets. A good layout effectively conveys the key features of a complex structure or system to a wide range of users. The primary goal of these types of methods is to optimize the arrangement of nodes such that strongly connected nodes appear close to each other.

The user is presented with a two-dimensional representation of multi-dimensional data, that is easy to understand and can be further investigated. We make use of Euclidean Minimal Spanning Tree (EMST) proposed by Stuetzle [72]. This skeleton forms the basic layout representation of the data. In order to compute the EMST, we need the high-dimensional point cloud in attribute space spanned by the entire set of attributes. Hence, for each position in the data set, an attribute vector that consists of individual multi-variate values is computed. A spanning tree connects all points in attribute space with line segments such that the resultant graph is connected and has no cycles. For a graph with n nodes, we end up with $n-1$ edges in the spanning tree. A spanning tree is an EMST if the sum of the euclidean distances between connected points is minimum. Since our graph layout follows a drill-down approach, each graph contains only a few thousands to tens of thousands of nodes. We apply Prim's algorithm [18] to compute the minimal spanning tree. For a graph with N nodes, $\frac{N(N-1)}{2}$ edges are chosen where each edge is given a weight by finding the euclidean distance between the pair of points. Our current approach is based on graph drawing, where the EMST is first projected to 2D by assigning each node an arbitrary position. Afterwards, the graph is laid out to achieve appropriate edge lengths and few edge intersections.

We choose the Fast Multipole Multilevel Method (FM3) method since it produces pleasing layouts and is relatively fast [30]. The basic approach of this algorithm tries to coarsen recursively an input graph G_0 to produce a series of smaller graphs $G_1 ... G_k$, until the size of coarsened graph falls below some threshold. We use the GPU implementation of a modified version of this algorithm by Godiyal et. al. [29]. Only the multipole expansion coefficients are considered and not the local expansion coefficients to approximate repulsive forces. These coefficients alone are sufficient to produce a high quality layout. Figure 3.7 give a perspective of the current algorithm.

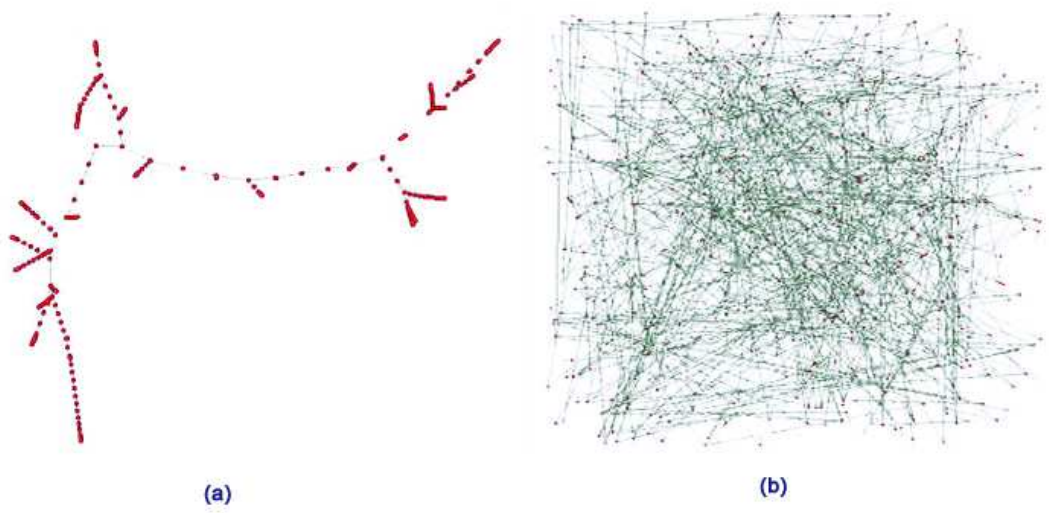

Figure 3.7 (a) Modified FM3 layout of EMST, (b) Randomly initialized EMST

3.3.2 Cluster Validity

Since we use an unsupervised approach of clustering, its often necessary to quantitatively evaluate the final partition. Cluster validity approaches based on relative criteria aim at finding the best clustering scheme that a clustering algorithm can define under certain assumptions and parameters. Here the basic idea is the evaluation of a clustering structure by comparing it to other clustering schemes which were produced by the same algorithm using different weight assignments. In this framework, we provide user an option of choosing from three indices. Namely,

- Davies-Bouldin index
- R-squared index
- SD validity index

Davies-Bouldin index: A similarity measure R_{ij}between the clusters C_i and C_j is defined based on a measure of dispersion of a cluster C_i and a dissimilarity measure between two clusters d_{ij}. The index satifies the following conditions

- $R_{ij} \geq 0$
- $R_{ij} = R_{ji}$
- if $s_i = 0$ and $s_j = 0$ then $R_{ij} = 0$
- if $s_i > s_k$ and $d_{ij} = d_{ik}$ then $R_{ij} > R_{ik}$
- if $s_i = s_k$ and $d_{ij} < d_{ik}$ then $R_{ij} > R_{ik}$

where s_i is the measure of dispersion of cluster C_i and d_{ij} the dissimilarity measure between two clusters. We choose

$$R_{ij} = (s_i + s_j)/d_{ij} \tag{3.3}$$

and hence the index is defined as

$$DB_{n_c} = \frac{1}{n_c} \sum_{i=1}^{n_c} R_i \tag{3.4}$$

$$R_i = \max_{i=1,...n_c, i \neq j} R_{ij}, i = 1, 2, ...n_c \tag{3.5}$$

We can notice that DB_{n_c} is the average similarity between each cluster c_i and its most similar one. We seek clusterings that minimize DB index as its desirable for clusters to exhibit minimum possible similarity to each other. This index exhibits no trends with respect to number of clusters and thus we seek to minimize its value in its plot versus number of clusters.

R-squared index: It is quite possible that hierarchical algorithms might be used to cluster the datasets. In such a case, this index proves to be much helpful in deciding the quality of clusters (note that this index can also be used for nonhierarchical algorithms like k-means). We run the clustering algorithm for different number of clusters each time and plot corresponding validity indices for these clusterings. We search for a "knee" in this graph. The number of clusters at which this "knee" is observed indicates a statistically optimal clustering for the dataset. The validity index takes the form

$$RS = \frac{SS_t - SS_w}{SS_t} \tag{3.6}$$

where SS means *Sum of Squares* and refers to

$$SS = \sum_{i=1}^{N} (X_i - \bar{X})^2 \tag{3.7}$$

and

- SS_w refers to sum of squares within group
- SS_b refers to sum of squares between group
- SS_t refers to total sum of squares, of entire dataset

RS of the new cluster is the ratio of SS_b over SS_t. SS_b is a measure of difference between groups. Since $SS_t = SS_b + SS_w$, the greater the SS_b the smaller the SS_w and vise versa. As a result, the greater the differences between groups are the more homogenous each group is and vise versa. Thus, RS may be considered as a measure of dissimilarity between clusters. Furthermore, it measures the degree of homogeneity between groups. The values of RS range between 0 and 1. In case that the value of RS is zero (0) indicates that no difference exists among groups. On the other hand, when RS equals 1 there is an indication of significant difference among groups.

SD validity index: Its definition is based on the concepts of average scattering for clusters and total seperation between clusters. Average scattering for clusters is defined as

$$Scatt(n_c) = \frac{1}{n_c} \sum_{i=1}^{n_c} \frac{\|\sigma(v_i)\|}{\|\sigma(X)\|} \tag{3.8}$$

and total seperation between clusters is

$$Dis(n_c) = \frac{D_{max}}{D_{min}} \sum_{k=1}^{n_c} \sum_{z=1}^{n_c} \|\sigma(v_k - v_z)\|^{-1} \tag{3.9}$$

where $D_{max} = max(\|v_i - v_j\|) \forall i, j \in 1, 2, ..n_c$ is the maximum distance between cluster centers. and $D_{min} = min(\|v_i - v_j\|) \forall i, j \in 1, 2, ..n_c$ is the minimum distance. SD index can now be defined as,

$$SD(n_c) = a.Scatt(n_c) + Dis(n_c) \tag{3.10}$$

where a is a weighing factor equal to Dis(C_{max}) where C_{max} is the maximum number of input clusters.

For in-depth description of other possible indices, please refer to [31]. As these indices are relative, we plot each of them as a line graph. Based on the index chosen, user searches for an optimum value denoting the best quality of clusters achieved over the process. Figure 3.8 denotes a plot of Davies-Bouldin index obtained for one of the experimental setups.

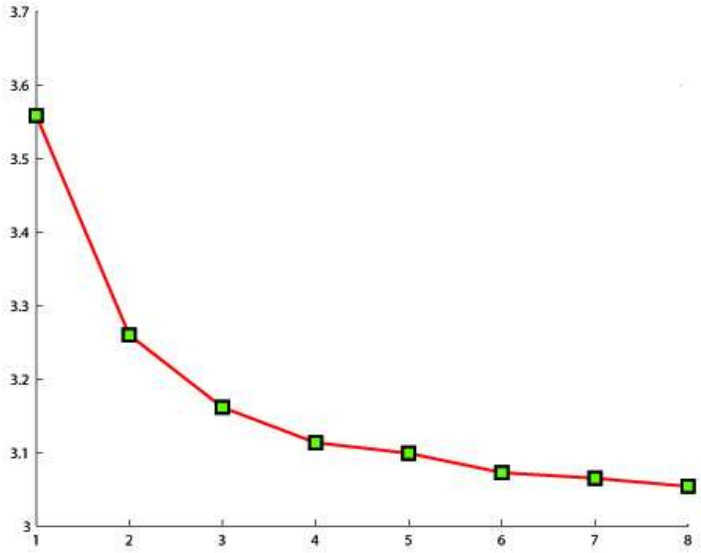

Figure 3.8 Davies-Bouldin index: With each iteration, the value decreases meaning better cluster quality

3.3.3 Interaction for Cluster Analysis

A zoom user interface allows user to change the scale of the viewer area. Holding the right mouse button down, a mouse left and a right makes the graphical display zoom out and zoom in respectively. This helps user pin down to a point of his/her interest.

User can rotate the display on either of the spatial axes (X, Y or Z) to change the view point. This is achieved by holding down the left mouse button and moving in any direction. User can translate the canvas by pressing middle mouse button and making a move in any direction. This provides great flexibility to analyse the structure of the graph layout and pin down to the interest area.

User can move the cursor over any valid data to browse through the details. This is a drill-down approach where visual words are displayed initially according to a graph layout algorithm. User can point the cursor over any node (each node represents a visual word) and click on it to generate a graph layout of underlying SIFT vectors assigned to that particular visual word. Another click on any node denoting a sift descriptor displays its interest region in an image as shown in Figure 3.9.

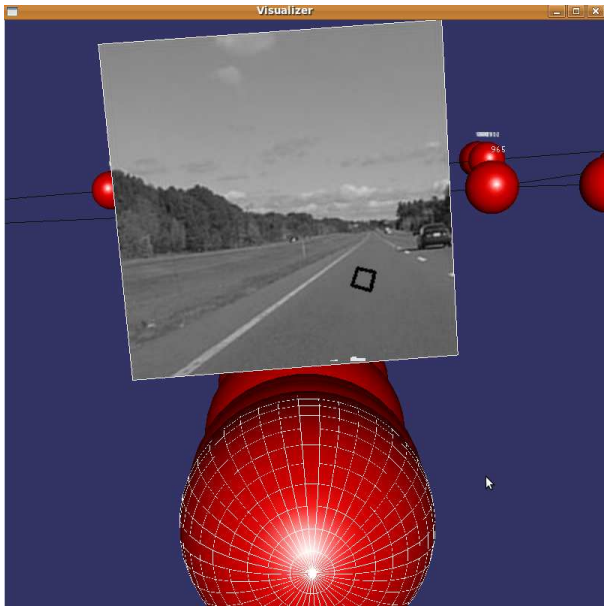

Figure 3.9 Drill-down to a sift descriptor in a cluster. The interest region is denoted by a black colored rectangle and the selected node is highlighted with a wired mesh enclosing it.

3.4 Automatic Weight Recommendation

It might often be tedious for a user to re-assign weights to each dimension based on cluster analysis. An automatic weight suggestion scheme proves handy in manual weight assignments. There are different ways of suggesting weights based on partition obtained from the clustering method. One way is to

choose

$$w_j = \begin{cases} 0 & \text{if } D_j = 0 \\ \frac{1}{\sum_{t=1}^{h} [\frac{D_j}{D_t}]^{\frac{1}{\beta-1}}} & \text{if } D_j \neq 0 \end{cases}$$

Where w_j is the attribute weight of j-th dimension, $\beta \leq 0$ or $\beta > 1$, h is the number of variables where $D_j \neq 0$ and

$$D_j = \sum_{l=1}^{k} \sum_{i=1}^{n} u_{i,l} d(x_{i,j}, z_{l,j})$$

$$\begin{cases} u_{i,l} = 1 & \text{if } \sum_{j=1}^{m} w_j^{\beta} d(x_{i,j}, z_{l,j}) \leq \sum_{j=1}^{m} {w_j}^{\beta} d(x_{i,j}, z_{t,j}) \\ & \text{for } 1 \leq \text{t} \leq \text{k} \\ u_{i,l} = 0 & \text{for t} \neq \text{l} \end{cases}$$

Z = $\{Z_1, Z_2, ...Z_k\}$ is a set of k vectors representing the centroids of the k clusters, X = $\{X_1, X_2, ...X_n\}$ is a set of n objects, each object $X_i = (x_{i,1}, x_{i,2}, ..., x_{i,m})$ is characterized by a set of m dimensions, U is a $n \times k$ partition matrix, $u_{i,l}$ is a binary variable and $u_{il} = 1$ indicates that object i is allocated to cluster l, d($x_{i,j}, z_{l,j}$) is the distance or dissimilarity measure between object i and centroid of cluster l on the j-th variable. This method of suggesting weights is a modified version from [34] where they aim to minimize an objective function

$$P(U, Z, W) = \sum_{l=1}^{k} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{i,l} w_j^{\beta} d(x_{i,j}, z_{l,j})$$

It is only a support process and the final decision to choose weights for each dimension is left to the user.

Chapter 4

Experiments

We show an application example of the framework with a collection of image classification categories. This dataset contains 4485 images with 15 different classes available with UIUC [2]. We apply SIFT feature detector on all images using Vedaldi's implementation [5]. We obtain close to 1.1 million SIFT vectors for this collection. We implemented the current tool using Qt, OpenSceneGraph [4] and OpenCV. All computations are performed on a commodity pc whose major configutation details are 2.4Ghz quad core processor, 2.5GB RAM and a Nvidia GTX 280 graphics card.

We use intersection kernel based Support Vector Machine (SVM) to build a classifier and a fast intersection kernel for testing [56]. However, the CPU computations we perform are not multi-threaded. We sample 60 SIFT descriptors from each image resulting in a total sample size of 0.26 million vectors approximately. We experiment with this framework based tool on individual classes as well as entire collection and the results are follows. After a series of iterations of weight re-assignment and clustering in tandem over sampled data with manual intervention, we observe that clusters formed are relatively stable, backed by a cluster validity index (for example, figure 3.8). Using the observed weights, we run a weighted kmeans on the entire class or dataset with respective K cluster centers.

4.1 Individual classes

There are approximately 250-270 images in each of the classes and sum up to around 16 thousand SIFT vectors. First, we assign weights based on 1D and 2D distributions to each of the dimensions. More uniformity means less weightage since there is less chance of finding clusters in the respective dimensions. We run weighted k-means on the sampled set using a pre-determined cluster size with 25 iterations. Once clustering is performed, we have a graphical look of clusters formed. Clusters which are similar must be closer to each other and vice-versa. A random sampled browsing through the minimum spanning tree of the clustering gives us a fair idea about the "visual" content inside. If we are satisfied with the clustering process (either visually or by cluster validity index), we can proceed to use those weights to cluster the entire collection. Else, we re-assign weights to each of the dimensions either by intuition or by mathematically suggested values based on data distribution. In these experiments,

we repeat this process from 3 to 10 times. Initial centers were the same for each cluster size (for comparision). The results for a few of the randomly picked classes are as follows.

No. of clusters	With uniform weights	By interactively adjusted weights
400	51.1	52.7
500	51.4	53.0
600	52.2	52.9
700	52.6	53.9
800	53.5	54.6
900	53.8	55.1
1000	54.8	55.9

Table 4.1 Results for ‘Mountain’

No. of clusters	With uniform weights	By interactively adjusted weights
400	48.7	50.9
500	51.5	53.3
600	51.1	53.3
700	52.3	54.1
800	52.8	53.6
900	53.6	53.9
1000	54.4	55.6

Table 4.2 Results for ‘Kitchen’

4.2 Sampled collection of all classes

As mentioned before, we end up with a total sample size of 0.26 million vectors approximately for entire collection of 15 different classes. We perform the same operations as in the previous experiments. We can notice that in figure 4.1, non-similar looking patches are clustered into the same visual word when given uniform weights.

The experimental results are as follows.

After using our tool, we could notice that similar looking patches did fall into the same visual words as shown in figure 4.2, 4.3

No. of clusters	With uniform weights	By interactively adjusted weights
400	51.7	55.3
500	52.5	55.2
600	53.1	54.9
700	53.7	55.3
800	54.2	55.6
900	54.1	55.8
1000	54.4	56.1

Table 4.3 Results for 'Highway'

No. of clusters	With uniform weights	By interactively adjusted weights
400	51.7	53.7
500	51.5	53.0
600	52.1	54.1
700	52.6	54.3
800	53.1	54.6
900	52.8	53.9
1000	53.4	55.2

Table 4.4 Results for 'All classes'

Over several experiments, we observe that dimensions, especially those belonging to the corner cells and corresponding to orientations $135°, 215°, 270°$ are assigned low weights by automatic suggestion schemas. Since we observe lower suggestive weights for a few dimensions in the previous iterative process, we redo the entire process once again. But this time, we discard a set of dimensions D_s = {4, 12, 22, 43, 44, 54, 55, 71, 78, 79, 83, 84, 110, 116} resulting around 11% decrement in data size. The observed overall classification accuracy in terms of percentage is shown in table 1.

Our CUDA based k-means algorithm takes half a second for an iteration for above sampled data. k-means is run upto convergence or 25 iterations, which ever is the earliest, there by consuming just over 12 seconds in a user interaction loop. Initial weight to each of the dimensions is assigned based on the behaviour observed in the histogram and PCP views. Once clusters are computed in a user interaction loop, automatic weights are suggested with $\beta = 7$ which takes less than half a second using a CUDA based implementation of method described in section 3.4. We consume about a minute to analyze the cluster quality and interactively adjust weights and proceed to the next round of clustering. In the above experiments, we loop over weight re-adjustment and clustering process eight times after

Figure 4.1 A few examples of SIFT descriptors in interest point regions denoted by black rectangular patches. All the current regions are grouped into the same visual word.

which a considerable drop in Davies-Bouldin index was not observed. After we conclude with a set of corresponding weights based on the sample data, our weighted k-means method is run over the entire dataset as an offline process. In order to avoid inconsistency, we generate 5 random sample sets and notify the mean classification accuracy obtained for entire dataset. Li et al. report a classification accuracy of 52.5% for SIFT with DoG [52] for the same dataset.

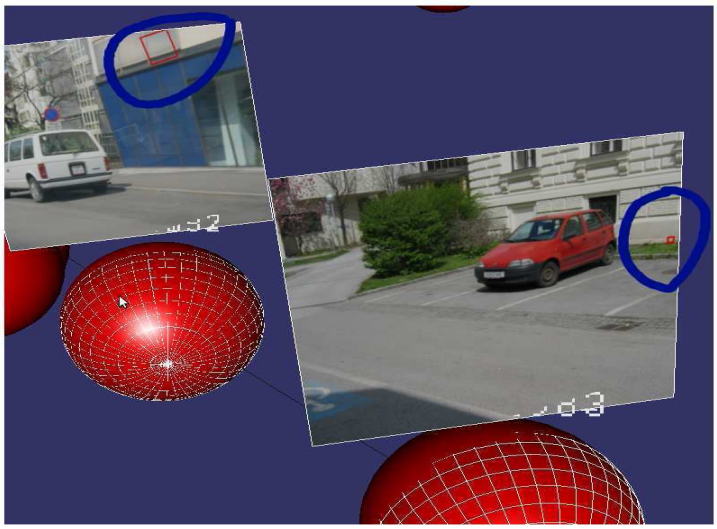

Figure 4.2 SIFT descriptors in interest point regions denoted by red rectangular patches. They are encircled with blue sketch to highlight their presence. We can observe that visually similar patches are closer after performing a weighted clustering.

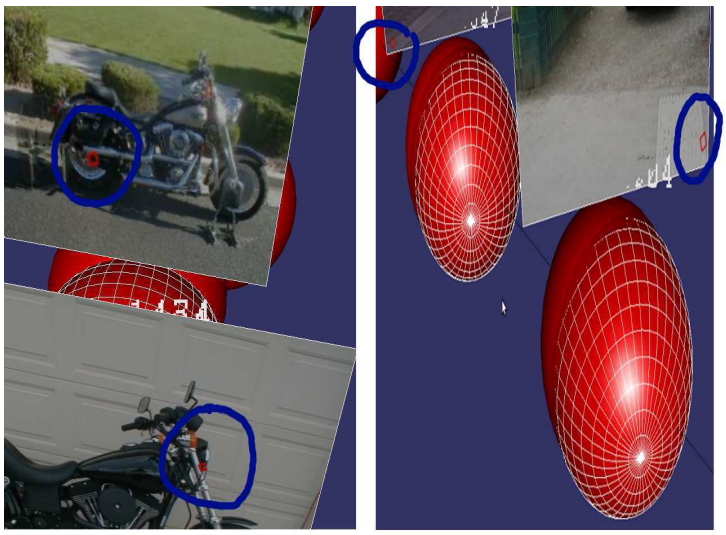

Figure 4.3 A few examples of SIFT descriptors in interest point regions denoted by red rectangular patches. They are encircled with blue markings to highlight their presence. The number of correctly mapped visually similar regions has increased after performing a user feedback based clustering.

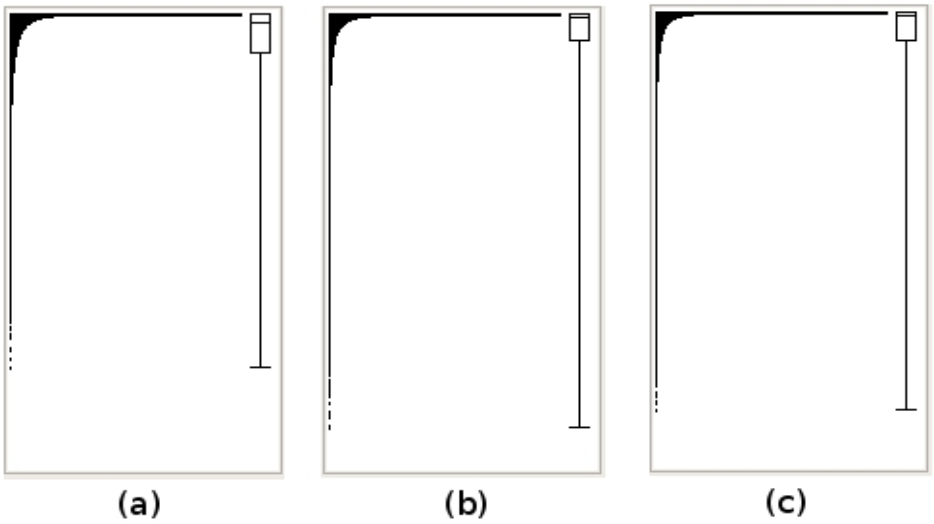

Figure 4.4 1D histogram corresponding to dimensions (a)84, (b) 110, (c) 124

Number of clusters	**Interactive Weights(all dimensions)**	**Interactive weights(discarded dimensions)**
400	53.7	55.1
500	53.0	54.7
600	54.1	55.7
700	54.3	56.6
800	54.6	56.9
900	53.9	56.3
1000	55.2	57.6

Table 4.5 Classificaton accuracy comparision.

Chapter 5

Conclusions

In this thesis, we consider very high dimensional data which is not spatially correlated and accounts for large storage space. Traditional techniques do not scale well with these kinds of datasets. Our attempt in the current case is to come up with a framework to improve the process of analysis and clustering of such datasets. Better clusters will result in generating robust indices thus leading to better results in accuracy. We provide a visualization based interactive tool using the proposed framework to guide the process of generating better clusters in the context of CBIR. We observed that distribution along 0 degree orientations is uniform and spread over a large range for several datasets. Though it takes considerable amount of time to analyze clusters, we can randomly sample a set of areas in the graph layout and observe the results which gives an overall view about the current process.

We notice that though this tool has been designed for analyzing SIFT vectors from a given set of images, we can extend it to any high-dimensional dataset which requires a cluster analysis scheme. We use SIFT descriptors with DoG to generate interest point descriptors. It would be interesting to see how it performs when SIFT is computed on a grid. The drawback is that it generates a lot of redundant data and so does the entire vector set. We have built the tool in such a way that any user can customize it using plugins. User has a new ranking scheme for his data or a new clustering method or another graph drawing method, he can incorporate it into this tool without any difficulty. The Parallel Coordinate Plots are rigid. We plan to make it interactive by providing a brush for subset comparision. User must be able to shift those vertical lines from one end of the display to the other and corresponding plot should be updated in near real-time. We like to explore the possibility of using other clustering methods to generate visual words and compare the performances. The graph drawing method which we use is a force based one. We look to incorporate other methods for analysis. The automatic weight suggestion method which we use is based on uniformity of data along a dimension. We look to explore other ways of suggesting weights in case user is not satisfied with the results. With encouraging results and many possibilities to improve upon the current framework/tool, we hope to come up with a better analysis technique which incorporates human feedback at the core of cluster generation.

Related Publications

Pavan Kumar Dasari and P. J. Narayanan, Interactive Visualization and Tuning of SIFT Indexing, Proceedings of the Vision, Modelling and Visualization Workshop 2010, Siegen, Germany, 97-105, Eurographics Association.

Bibliography

[1] http://searchenginewatch.com/article/2068075/search-engine-sizes.

[2] http://www-cvr.ai.uiuc.edu/ponce%20grp/data/index.html.

[3] http://www.developer.nvidia.com/cuda.

[4] http://www.openscenegraph.org/projects/osg.

[5] http://www.vlfeat.org/.

[6] http://www.worldwidewebsize.com/.

[7] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, SIGMOD '98, pages 94–105, New York, NY, USA, 1998. ACM.

[8] A. O. Artero, M. C. F. de Oliveira, and H. Levkowitz. Uncovering clusters in crowded parallel coordinates visualizations. In *INFOVIS '04: Proceedings of the IEEE Symposium on Information Visualization*, pages 81–88, 2004.

[9] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[10] G. Battista. *Graph drawing: algorithms for the visualization of graphs*. Prentice Hall, 1999.

[11] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5:537–550, 1994.

[12] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:509–522, 2002.

[13] A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *ARTIFICIAL INTELLIGENCE*, 97:245–271, 1997.

[14] F.-J. Brandenburg. Nice drawings of graphs are computationally hard. In *Selected Contributions from the on 7th Interdisciplinary Workshop on Informatics and Psychology: Visualization in Human-Computer Interaction*, pages 1–15, London, UK, 1990. Springer-Verlag.

[15] R. Bürger, P. Muigg, M. Ilcik, H. Doleisch, and H. Hauser. Integrating local feature detectors in the interactive visual analysis of flow simulation data. In A. Y. K. Museth, T. Möller, editor, *Proceedings of Eurographics/ IEEE-VGTC Symposium on Visualization 2007*, pages 171–178, 2007.

[16] P. Carbonetto, N. de Freitas, P. Gustafson, and N. Thompson. Bayesian feature weighting for unsupervised learning, with application to object recognition. *Workshop on Artificial Intelligence and Statistics*, jun 2003.

[17] R. Caruana and D. Freitag. Greedy attribute selection. In *In Proceedings of the Eleventh International Conference on Machine Learning*, pages 28–36. Morgan Kaufmann, 1994.

[18] D. Cherition and R. E. Tarjan. Finding minimum spanning trees. *SIAM journal on computing*, 5:724–741, 1976.

[19] H. Chernoff. The Use of Faces to Represent Points in K-Dimensional Space Graphically. *Journal of the American Statistical Association*, 68(342):361–368, 1973.

[20] H. D. Ueber die stetige abbildung einer line auf ein flachensta 1/4ck. *Mathematische Annalen*, 3(38):459–, 1891.

[21] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 886–893, 2005.

[22] M. Dash and H. Liu. Feature selection for clustering. In *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*, PADKK '00, pages 110–121, London, UK, 2000. Springer-Verlag.

[23] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40(2):1–60, 2008.

[24] M. Devaney and A. Ram. Efficient feature selection in conceptual clustering. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 92–97, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[25] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:891–906, 1991.

[26] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55:119–139, August 1997.

[27] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Softw. Pract. Exper.*, 21:1129–1164, November 1991.

[28] Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. In *VIS '99: Proceedings of the conference on Visualization '99*, pages 43–50. IEEE Computer Society Press, 1999.

[29] A. Godiyal, J. Hoberock, M. Garland, and J. C. Hart. Rapid multipole graph drawing on the gpu. *Graph Drawing: 16th International Symposium, GD 2008, Heraklion, Crete, Greece, September 21-24, 2008. Revised Papers*, pages 90–101, 2009.

[30] S. Hachul and M. Jünger. Large-graph layout with the fast multipole multilevel method. Technical report, Zentrum für Angewandte Informatik Köln, Lehrstuhl Jünger, December 2005.

[31] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. Cluster validity methods: part i, ii. *SIGMOD Rec.*, 31, 2002.

[32] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. Clustering validity checking methods: part ii. *SIGMOD Rec.*, 31(3):19–27, 2002.

[33] M. Hilbert and P. Lpez. The worlds technological capacity to store, communicate, and compute information. *Science*, 331(6018):692–693, 2011.

[34] J. Z. Huang, M. K. Ng, H. Rong, and Z. Li. Automated variable weighting in k-means type clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(5):657–668, 2005.

[35] A. Inselberg, M. Reif, and T. Chomut. Convexity algorithms in parallel coordinates. *J. ACM*, 34:765–801, October 1987.

[36] B. K. J. M. Chambers, W. S. Cleveland and P. A. Tukey. *Graphical Methods for Data Analysis*. Chapman and Hall, New York, 1983.

[37] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[38] J. Johansson, M. Cooper, and M. Jern. 3-dimensional display for clustered multi-relational parallel coordinates. In *IV '05: Proceedings of the Ninth International Conference on Information Visualisation*, pages 188–193, 2005.

[39] J. Johansson, P. Ljung, M. Jern, and M. Cooper. Revealing structure within clustered parallel coordinates displays. In *INFOVIS '05: Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, page 17, 2005.

[40] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:506–513, 2004.

[41] D. Keim, H.-P. Kriegel, and M. Ankerst. Recursive pattern: a technique for visualizing very large amounts of data. In *Visualization, 1995. Visualization '95. Proceedings., IEEE Conference on*, pages 279 –286, 463, oct-3 nov 1995.

[42] D. A. Keim. Pixel-oriented visualization techniques for exploring very large databases. *Journal of Computational and Graphical Statistics*, 5:58–77, 1996.

[43] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, 2002.

[44] Y. Kim, W. N. Street, and F. Menczer. Feature selection in unsupervised learning via evolutionary search. In *In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 365–369. ACM Press, 2000.

[45] K. Kira and L. A. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *AAAI*, pages 129–134. AAAI Press and MIT Press, 1992.

[46] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97:273–324, December 1997.

[47] J. Komorowski, L. Polkowski, and A. Skowron. Rough sets: A tutorial. In *Rough-Fuzzy Hybridization: A new method for decision making*. Singapore: Springer-Verlag, 1998.

[48] I. Kononenko. Estimating attributes: Analysis and extensions of relief. pages 171–182. Springer Verlag, 1994.

[49] N. Kwak and C.-H. Choi. Input feature selection by mutual information based on parzen window. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(12):1667 – 1671, dec 2002.

[50] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using affine-invariant regions. In *In Proc. CVPR*, pages 319–324, 2003.

[51] M. D. Lee, M. A. Butavicius, and R. E. Reilly. An empirical evaluation of chernoff faces, star glyphs, and spatial visualizations for binary data. In *In APVis 03: Proceedings of the Asia-Pacific symposium on Information visualisation*, pages 1–10. Australian Computer Society, Inc, 2003.

[52] F.-F. Li and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 524–531. IEEE Computer Society, 2005.

[53] X. Li and B. Furht. Design and implementation of digital libraries. pages 415–450, 2000.

[54] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 1150, 1999.

[55] D. G. Lowe. Local feature view clustering for 3d object recognition. *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01)*, I:682–688, 2001.

[56] S. Maji, A. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –8, june 2008.

[57] O. A. McBryan. Genvl and wwww: Tools for taming the web. In *First International World Wide Web Conference*, 1994.

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

[59] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.

[60] P. Mitra, C. Murthy, and S. Pal. Unsupervised feature selection using feature similarity. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(3):301 –312, mar 2002.

[61] S. Moen. Drawing dynamic trees. *IEEE Software*, 7:21–28, July 1990.

[62] P. Muigg, J. Kehrer, S. Oeltze, H. Piringer, H. Doleisch, B. Preim, and H. Hauser. A four-level focus+context approach to interactive visual analysis of temporal features in large scientific data. *Computer Graphics Forum*, 27(3):775–782, 2008.

[63] J. Pena, J. Lozano, P. Larranaga, and I. Inza. Dimensionality reduction in unsupervised learning of conditional gaussian networks. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(6):590 –603, jun 2001.

[64] M. I. Petrovskiy. Outlier detection algorithms in data mining systems. *Program. Comput. Softw.*, 29(4):228–237, 2003.

[65] P. Pudil, J. Novovičová, and J. Kittler. Floating search methods in feature selection. *Pattern Recogn. Lett.*, 15:1119–1125, November 1994.

[66] G. G. Robertson, J. D. Mackinlay, and S. K. Card. Cone trees: animated 3d visualizations of hierarchical information. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, CHI '91, pages 189–194, New York, NY, USA, 1991. ACM.

[67] V. Roth and T. Lange. Feature selection in clustering problems. In *In Advances in Neural Information Processing Systems 16*. MIT Press, 2003.

[68] M. Sahami. *Using machine learning to improve information access*. PhD thesis, Stanford, CA, USA, 1999. AAI9924490.

[69] P. Sand and A. W. Moore. Repairing faulty mixture models using density estimation. In *In Proceedings of the 18th International Conf. on Machine Learning*, pages 457–464. Morgan Kaufmann, 2001.

[70] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, Oct. 2003.

[71] J. T. Stasko and E. T. Kraemer. A methodology for building application-specific visualizations of parallel programs. In *GVU Technical Report;GIT-GVU-92-10*. Georgia Institute of Technology.

[72] W. Stuetzle. Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample. *J. Classification*, 20(1):025–047, 2003.

[73] L. Talavera. Dependency-based feature selection for clustering symbolic data. *Intell. Data Anal.*, 4:19–28, January 2000.

[74] S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*, pages 107–118, New York, NY, USA, 2001. ACM.

[75] L. A. Treinish. Task-specific visualization design: a case study in operational weather forecasting. In *VIS '98: Proceedings of the conference on Visualization '98*, pages 405–409, Los Alamitos, CA, USA. IEEE Computer Society Press.

[76] C.-L. Tsang, M.-W. Mak, and S.-Y. Rung. Cluster-dependent feature transformation with divergence-based out-of-handset rejection for robust speaker verification. In *Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia.*, volume 3, pages 1399–1403, dec. 2003.

[77] S. Vaithyanathan and B. Dom. Generalized model selection for unsupervised learning in high dimensions. In *Proceedings of Neural Information Processing Systems*, pages 970–976. MIT Press, 1999.

[78] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511 – I–518 vol.1, 2001.

[79] V. Šaltenis and J. Aušraite. Data visualization: ideas, methods, and problems. *Informatics in education*, 1(1):129–148, 2002.

[80] M. O. Ward. Xmdvtool: integrating multiple methods for visualizing multivariate data. In *VIS '94: Proceedings of the conference on Visualization '94*, pages 326–333, 1994.

[81] R. E. Wilson. The impact of software on crime mapping. *Social Science Computer Review*, 25(2):135–142, 2007.

[82] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. *Intelligent Systems and their Applications, IEEE*, 13(2):44 –49, mar/apr 1998.