

University of Louisville

ThinkIR: The University of Louisville's Institutional Repository

Electronic Theses and Dissertations

8-2010

Adaptive constrained clustering with application to dynamic image database categorization and visualization.

Jason Daron Meredith 1982-
University of Louisville

Follow this and additional works at: <https://ir.library.louisville.edu/etd>

Recommended Citation

Meredith, Jason Daron 1982-, "Adaptive constrained clustering with application to dynamic image database categorization and visualization." (2010). *Electronic Theses and Dissertations*. Paper 965.
<https://doi.org/10.18297/etd/965>

This Doctoral Dissertation is brought to you for free and open access by ThinkIR: The University of Louisville's Institutional Repository. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of ThinkIR: The University of Louisville's Institutional Repository. This title appears here courtesy of the author, who has retained all other copyrights. For more information, please contact thinkir@louisville.edu.

**ADAPTIVE CONSTRAINED CLUSTERING WITH
APPLICATION TO DYNAMIC IMAGE DATABASE
CATEGORIZATION AND VISUALIZATION**

By

Jason Daron Meredith
B.S., CECS, University of Louisville, 2004
M.Eng., CECS, University of Louisville, 2005

A Dissertation
Submitted to the Faculty of the
Graduate School of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

August 2010

Adaptive Constrained Clustering with Application to Dynamic Image Database Categorization and Visualization

By

Jason Daron Meredith
B.S., CECS, University of Louisville, 2004
M.Eng., CECS, University of Louisville, 2005

A Dissertation Approved On

05 / 05 / 2010
Date

by the following Dissertation Committee:

Dissertation Director

ACKNOWLEDGEMENTS

I would like to thank the many people who made my time at the University of Louisville memorable.

First, I would like to thank my advisor Dr. Hichem Frigui, for his guidance, support, and patience during my time in the Ph.D program. I would also like to thank the members of my committee, Dr. Dar-Jen Chang, Dr. Ahmed Desoky, Dr. Carol Hanchette, and Dr. Rammohan Ragade. Finally, I would like to thank my family and friends to whom I am exceedingly grateful for their constant interest and encouragement towards my work.

ABSTRACT

ADAPTIVE CONSTRAINED CLUSTERING WITH APPLICATION TO DYNAMIC IMAGE DATABASE CATEGORIZATION AND VISUALIZATION

Jason Daron Meredith

August 9, 2010

The advent of larger storage spaces, affordable digital capturing devices, and an ever growing online community dedicated to sharing images has created a great need for efficient analysis methods. In fact, analyzing images for the purpose of automatic categorization and retrieval is quickly becoming an overwhelming task even for the casual user.

Initially, systems designed for these applications relied on contextual information associated with images. However, it was realized that this approach does not scale to very large data sets and can be subjective. Then researchers proposed methods relying on the content of the images. This approach has also proved to be limited due to the semantic gap between the low-level representation of the image and the high-level user perception.

In this dissertation, we introduce a novel clustering technique that is designed to combine multiple forms of information in order to overcome the disadvantages observed while using a single information domain. Our proposed approach, called Adaptive Constrained Clustering (ACC), is a robust, dynamic, and semi-supervised

algorithm. It is based on minimizing a single objective function incorporating the abilities to: (i) use multiple feature subsets while learning cluster independent feature relevance weights; (ii) search for the optimal number of clusters; and (iii) incorporate partial supervision in the form of pairwise constraints. The content of the images is used to extract the features used in the clustering process. The context information is used in constructing a set of appropriate constraints. These constraints are used as partial supervision information to guide the clustering process. The ACC algorithm is dynamic in the sense that the number of categories are allowed to expand and contract depending on the distribution of the data and the available set of constraints.

We show that the proposed ACC algorithm is able to partition a given data set into meaningful clusters using an adaptive, soft constraint satisfaction methodology for the purpose of automatically categorizing and summarizing an image database. We show that the ACC algorithm has the ability to incorporate various types of contextual information. This contextual information includes: spatial information provided by geo-referenced images that include GPS coordinates pinpointing their location, temporal information provided by each image’s time stamp indicating the capture time, and textual information provided by a set of keywords describing the semantics of the associated images.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
LIST OF TABLES	x
LIST OF FIGURES	xi

CHAPTER

I	INTRODUCTION	1
A	Motivations	1
B	Contributions	4
C	Dissertation Overview	5
II	LITERATURE SURVEY	6
A	Prototype-Based Clustering Algorithms	7
1	K-Means Algorithm	8
2	The Fuzzy C-Means (FCM) Algorithm	9
3	The Gustafson-Kessel (GK) Algorithm	10
B	Determining the Optimal Number of Clusters	11
1	Validity-Based Approaches	12
2	Objective Function Based Approach	13
C	Feature Selection and Weighling	15
D	Semi-Supervised Clustering	17
1	The K-Means Algorithm with Pairwise Constraints	19

2	Pairwise-Constrained Competitive Agglomeration	22
III	ADAPTIVE CONSTRAINED CLUSTERING	26
A	The Constrained Clustering (CC) Algorithm	26
B	The Adaptive Constrained Clustering (ACC) Algorithm	34
C	Constraint Selection	38
D	Computational Complexity	39
E	Convergence Properties	39
IV	EXPERIMENTAL EVALUATIONS	41
A	Introduction	41
1	Data Sets	41
2	Performance Measures	42
B	Experimental Results	44
C	Convergence Properties and Computational Complexity	47
V	DYNAMIC IMAGE DATABASE CATEGORIZATION AND VISUALIZATION USING ADAPTIVE CONSTRAINED CLUS- TERING	54
A	Motivations	54
B	Content-Based Image Database Categorization	55
1	Feature Extraction	55
2	Image Database Categorization Using Machine Learning Techniques	57
3	Image Database Categorization using the ACC algorithm	59
C	Constraint Selection	61
1	Active Selection of Constraints	62

2	Exploring the Unsatisfied Should-not links to find the optimal number of clusters	63
D	Categorization using the ACC with Constraints derived from Spatial Information	64
1	Cost of Violating Spatial Constraints	65
2	Spatial Constraints Construction	66
3	Experimental Results using Geo-referenced Data	67
E	Categorization using the ACC with Constraints derived from Temporal Information	73
1	Cost of Violating Temporal Constraints	76
2	Temporal Constraints Construction	77
3	Categorization using the ACC with Constraints derived from Textual Information	80
4	Cost of Violating Textual Constraints	81
5	Textual Constraints Construction	81
F	Experimental Evaluation and Comparison	82
1	Image Collection	85
2	Incorporating Constraints	85
3	Objective Evaluation	86
4	Subjective Evaluation	93
VI	CONCLUSIONS AND FUTURE WORK	103
	REFERENCES	108
	Appendices	116
A	Creation of Synthetic Data Sets	117

LIST OF TABLES

TABLE	Page
1 Data sets used in the algorithm comparison.	42
2 Contingency table.	43
3 Feature relevance weights learned during initialization of the ACC algorithm.	45
4 The observed running time for each algorithm on sample Data set 1. .	52
5 The observed running time for each algorithm on sample Data set 2. .	53
6 Similar images from different geographical locations	71
7 Images with similar contextual semantics and differing content information	86
8 The observed running time for each algorithm on sample data set in Section F.1.	89
9 The observed running time of an example application of dynamic image categorization using the ACC algorithm.	92
10 Validity scores for subjective evaluations.	97

LIST OF FIGURES

FIGURE		Page
1	An illustrative example of over-agglomeration. (a) A 2-D data set with two clusters. (b) The ground truth of the two clusters. (c) Initialization of the CA algorithm with a over estimation of clusters $C_{max} = 9$. (d) After a few iterations, the number of clusters has reduced to 6, where the agglomeration merges two clusters inconsistently with the ground truth. (f) Upon convergence, portions of two distinct clusters have merged, yielding unfavorable clustering results. The CA algorithm cannot increase the number of clusters and recover from this local minima.	35
2	Illustration of the cluster splitting process in the ACC algorithm. (a) Partition resulting from the CA algorithm and selection of a <i>should-not</i> link constraint. (b) Splitting of the black cluster to satisfy the should not link constraint. (c) Final partition where the ACC has recovered from the over-agglomeration.	38
3	The synthetic data sets used to evaluate and compare the algorithm. Points from each cluster are represented by a different color and symbol.	42
4	Initialization of Data set 1. The shape of each symbol refers to the contextual meaning of each point, and the color represents the assigned cluster.	45
5	Intermediate results of the PCCA algorithm on Data Set 1	46
6	Intermediate results of the ACC algorithm on Data Set 1	47

7	Performance evaluations for Data Set 1	48
8	Initialization of Data Set 2. The shape of each symbol refers to the contextual meaning of each point, and the color represents the assigned cluster.	48
9	Intermediate results of the PCCA Algorithm on Data Set 2	49
10	Intermediate results of the ACC algorithm on Data Set 2	49
11	Performance evaluations for Data Set 2	50
12	Evaluation of the objective function over 100 synthetic trials. (a) Results for the SCADCA algorithm (b) Results for the PCCA algorithm (c) Results for the ACC algorithm	51
13	The number of clusters used per iteration in the ACC algorithm. . . .	52
14	Average and standard deviation using the Q_{Rand} , Q_{Jacc} , and Q_{FMI} performance measures over 100 runs with different initialization. The low variance of the ACC indicates less sensitivity to different initializations.	53
15	Illustrative example of cluster zooming. (a) An overview of the images in the data set. (b) The images contained in the cluster represented by the flower. (c) Images contained in a subsequent zoom level, continuing to navigate based on the selection of flowers.	58
16	Sample data set demonstrating the need for partial supervision. (a) Known truth values for given data set. Marker color indicates cluster assignment, marker shape indicates class label. (b) Results of a typical unsupervised clustering algorithm. Note the misclassification of points in the overlapping region between clusters.	59

17	Demonstration of the use of pairwise constraints. (a) Enlarged view of the overlap in clusters from Figure 16(b). (b) Resulting constraints constructed from known information in Figure 16(a). (c) Reclustering with consideration to the constraints allows the algorithm to correctly partition the data set.	60
18	Comparison of a map cluttered with images versus a map with clusters of images.	65
19	Calculation of the rho function.	66
20	Spatial layout of a subset of images tagged with spatial coordinates Each region corresponds to a distinct area across the globe.	68
21	Layout of the images in Fig. 20 in the feature space. The image borders represent the three different geographical regions.	69
22	Application of spatial constraints. (a) An abstract view of the images in Fig 20. (b) Results generated by a typical unsupervised clustering algorithm. (c) A set of constraints are selected between points that are in different regions but should and should not be in the same cluster. (d) Partition generated with the consideration of the constraints where the number of clusters is expanded to four.	70
23	Illustrative example of spatial region expansion. (a) Overview of the image collection with representative clusters. (b) Reorganized data representing the USA region. (c) City level representation of data from selected region. (d) All images from street level region.	72
24	Unsupervised clustering results. (a) Results showing visually similar images from two distinct regions. (b) Results showing visually similar images from three distinct regions.	74

25	Viewing images contained in a cluster of interest. (a) Cluster containing images from Louisville. (b) Cluster containing images from surrounding Louisville area.	75
26	Viewing Cluster results from the Paris region.	76
27	Sample cluster based on visual content.	78
28	Layout of Fig. 27 based on temporal information	78
29	Application of temporal constraints. (a) A set of should link constraints between images with like time signatures. (b) A set of should not link constraints between images with differing time signatures.	79
30	Sample clusters generated by the ACC algorithm using the temporal constraints in Fig. 29	79
31	Sample data set for textually constrained clustering.	83
32	Sample clustering of images in Fig. 31 clustered using image content. (a) Images of sunsets with the sun under the horizon. All images tagged with the keyword "sunset". (b) Various images with a prominent orange coloring and associated image tags. (c) Cluster containing red images with assigned annotations. (d) Partition containing nighttime images of people.	83
33	Application of textual constraints. (a) A set of should link constraints between images that share the keyword "sunset". (b) A set of should not link constraints between images with no words in common. . . .	84
34	Sample clustering displaying enhanced semantics from textual constraints.	84
35	Average cluster purity of the partitions generated by the ACC, SCADCA, and PCCA algorithms.	88
36	Percentage of satisfied constraints from objective experiments.	88

37	Evaluation of the objective function over 100 sample trials. (a) Results for the SCADCA algorithm (b) Results for the PCCA algorithm (c) Results for the ACC algorithm	90
38	Illustrative example of spatial region expansion. (a) Overview of the image collection with representative clusters initially clustered in an off-line fashion. (b) Reorganized data representing the user's preference, processed online. (c) The images contained in the area of interest are dynamically reclustered with respect to the current region. (d) At the street level, the number of images contained the region do not require categorization and are displayed in their entirety.	91
39	Comparison of clusters from the ACC and SCADCA algorithm. These results illustrate the benefit of using partial supervision.	93
40	Comparison of ACC and PCCA clusters illustrating the benefits of using relevance feature weights.	93
41	Results comparing the ACC and the PCCA algorithm, demonstrating increased clustering performance.	94
42	Comparing the ACC and the PCCA algorithm, where the increased clustering performance can be attributed to increased constraint satisfaction.	94
43	Results from the ACC algorithm indicating the presence of <i>should</i> link constraints.	95
44	An example of user subjectivity.	96
45	A screen shot of the subjective evaluation test.	97
46	Validity results from subjective evaluation.	98
47	Clusters utilized for subjective evaluation, where the partial supervision information provides an increase in cluster purity.	99

48	Using constraints, the PCCA and ACC algorithms are able to increase purity compared the SCADCA.	100
49	Sample result set, demonstrating the issue of subjectivity.	101
50	Illustrative example of subjective clusters showing forest scenes	102
51	Subjective clusters where a strong theme may not be present.	102

CHAPTER I

INTRODUCTION

A Motivations

Moore's law is a predictive trend that describes a long-term trend in computer hardware in which the number of transistors that can be placed inexpensively on a transistor doubles approximately every two years [1]. Following this trend, advancements in technology have allowed for an amazing ease of capture, storage, and sharing options in digital imagery. Societal web sites, such as Flickr [2] and Panoramio [3], contain collections of images numbering the millions. While it is apparent that examples of expansive and expanding image data sets exist, tasks such as navigation, retrieval, or categorization of these data sets is becoming rapidly overwhelming. Therefore, research directions exploring the ability to perform these tasks while automatically processing images searching for trends in similar images is becoming very active.

Image retrieval techniques originated as text-based methods in the 1970s. Typical methods relied on manual annotations associated with each image to transform the problem into a standard text retrieval process [4]. Despite this straightforward and efficient approach, along with the numerous advancements in text-based retrieval techniques [5], numerous issues remain difficult to solve. For example, text-based methods rely on complete and accurate annotations for each image using a predetermined vocabulary [6]. In reference to images, this implies that each image would need to be manually annotated which, with respect to a rapidly expanding data set,

can very quickly become an expensive, time-consuming task. These methods also suffer from the issue of user subjectivity. Inconsistencies in the use of terminology and assignments between indexers, variability between terms that describe the same subject (synonyms), and describing different subjects (ambiguity) are just a few examples of user subjectivity that can adversely affect the performance of text-based image retrieval techniques.

The combination of rapidly growing, large-scale image collections and the difficulties presented by manually annotated text-based techniques led to a new trend in retrieval methods during the 1990s. The new methods relied on using the visual properties of images to perform retrieval [7, 8, 9, 10, 11, 12]. Color, texture, shape, and spatial layout are a few examples of the image content features that can be extracted and used for the purpose of indexing. Similarity between images is determined by using appropriate distance measures in conjunction with sets of these low-level features. Then, images are retrieved based on the similarity to a given query image. In recent years, many Content-Based Image Retrieval (CBIR) systems have been developed [13, 14, 15, 16, 17, 18]. Several of these systems employ a query-by-example methodology for retrieval purposes, utilizing various indexing features in either the *image* or the *region* domain. The major drawback of the CBIR framework is what is known as the *semantic gap*. The semantic gap is defined as the difficulty of inferring high-level semantic meaning from the extracted low-level features of an image [19]. The semantic gap has led to a severe limitation in the advancement of CBIR systems in real applications.

Research directions in recent years have focused on methods for bridging the semantic gap. These methods use various forms of relevance feedback [20, 21, 22, 23, 24] and the inclusion of textual keywords that can be extracted in an unsupervised or semi-supervised manner [25, 26, 27, 28]. The combination of visual features with

textual descriptors proved to be effective in overcoming the drawbacks of each independent system. For example, in [29] textual keywords are used to reduce the search space for content-based methods. Unfortunately this method still suffers from the primary drawbacks of text-based image retrieval, cost and coverage. Therefore, methods that combine textual keywords and image content may only provide a partial solution to the semantic gap problem.

Approaches that combine image content with non-visual descriptors can exist without the use of textual keywords [30]. It can be theorized that one of the primary contributors to the issue of semantic gap is the fact that the image capture and image analysis processes are isolated in both time and space. This is in spite of the ability of image capturing devices to provide several clues to both of these pieces of contextual information. For instance, using a combination of temporal and spatial context as side information could provide semantically meaningful information aiding in image analysis [31]. Current image capturing devices have the ability to efficiently store and provide a number of contextual metadata relevant to the point of capture. This method can provide image analysis techniques with the ability to use information that was previously lost between the point of capture and the moment of analysis. As an example, consider the use of GPS coordinates that are embedded in the image, to create *geo*-referencing of where each image was taken.

As the trend of increased online socialization and the trend characterized by Moore's Law create rapidly growing image collections and the resources to store them, there will be an ever increasing need for image retrieval methods and the need for other image analysis techniques for media management.

Using statistical learning methods, image database categorization methods attempt to group images into semantically meaningful categories using their low-level features. These categorizations could be used to index an image database and provide

means to navigate through it.

B Contributions

In this dissertation, we propose a robust approach to automatically categorize and summarize an image database, using low level visual features and high-level contextual information. With regards to human to human interaction, the recognition of objects within images can be difficult without context. Thus, the combination of contextual and visual information is vital when trying to create a semantically meaningful understanding. Overcoming the semantic gap problem in image summarization and categorization methods is consequently more likely to be successful when context, derived from the images, is used along with the content of the images.

Our approach is based on a new dynamic, semi-supervised clustering algorithm. The algorithm is designed to overcome several issues that affect the performance of traditional clustering algorithms. We formulate a single objective function that combines unsupervised learning, semi-supervised learning, competitive learning, and feature discrimination. The resulting algorithm was used to categorize a large collection of images. Low-level image content features, from the MPEG-7 standard [32], were used to describe the images and define the feature space. High-level contextual information was used, as semi-supervised information, to guide the clustering. The semi-supervised information was formulated as a set of constraints that suggest which instances should or should not reside in the same cluster. For example, spatial information could be used to discover clusters of similar images that are close to each other geographically. Similarly, temporal information could assist the clustering by respecting the dates and times images were taken and the length of time between them. Textual information, on the other hand, could be used to guide the algorithm towards finding clusters that share common keywords and thus, are semantically more

meaningful.

C Dissertation Overview

The remainder of the dissertation is organized as follows: Chapter II provides a literature review of related work relevant to our proposed method. Chapter III presents our new approach to clustering and feature discrimination using partial semi-supervision to dynamically and semantically combine context and content information. In Chapter IV, we provide experimental evaluations of our approach compared to existing methods. Chapter V provides an overview of image database categorization using our approach demonstrating various methods of constraint creation and validation of our experimental results. Finally, Chapter VI outlines our conclusions and potential future work that could be researched in order to expand the functionality of the proposed algorithm.

CHAPTER II

LITERATURE SURVEY

Clustering is defined as the partitioning of data into groups that share some common traits. Traits, or features, are considered to be common based on some form of distance or similarity measure. In this chapter, we provide an overview of several clustering methodologies and algorithms relevant to our proposed approach.

Numerous clustering approaches exist, most of which can be divided into three categories; hierarchical, density-based, and partitional clustering. Hierarchical clustering methods [33, 34, 35] partition data by obtaining a nested sequence based on a graphical representation known as a dendrogram. Methods based on the density-based approach [36, 37, 15, 38, 39] use local properties of the data objects for grouping purposes. Partitional, also known as prototype-based, methods minimize an objective function and create a single partition [40, 41, 42, 25, 26]. Prototype-based clustering methods provide several advantages when compared to the other methods. In this approach, points are allowed to dynamically shift from one cluster to another. They also provide the ability to incorporate knowledge obtained about cluster shapes and sizes in conjunction with appropriate prototypes and distance measures in their objective functions.

The subsequent sections of this chapter are arranged as follows: Section A reviews a number of prototype-based clustering methods. Methods dedicated to searching for the optimal number of clusters are covered in Section B. An overview of feature selection approaches is given in Section C. Finally, Section D presents an introduc-

tion to semi-supervised clustering, which is a search-based approach to partitioning the data where user-provided constraints or labels are used to guide the clustering process.

A Prototype-Based Clustering Algorithms

Prototype-based clustering methods attempt to find an optimal partition of a data set by minimizing an objective function. The assignment criterion can be described as either hard (crisp) or soft (fuzzy) depending on whether each point belongs to one cluster exclusively or to multiple clusters with varying degrees. In general, fuzzy algorithms perform better than crisp because of their reduced tendency to get trapped in local minima, and their ability to provide a better description of the data.

Let $\mathcal{X}=\{\mathbf{x}_j \in \mathbb{R}^p | j=1, \dots, N\}$ be a set of N feature vectors in a p -dimensional feature space. Let $\mathbf{B}=(\beta_1, \dots, \beta_c)$ represent a C -tuple of prototypes each of which characterizes one of the C clusters. Each β_i consists of a set of parameters, such as a center and a covariance matrix. Let u_{ij} represent the membership of \mathbf{x}_j in cluster β_i . For the crisp case, the $C \times N$ binary C -partition, $\mathbf{U}=[u_{ij}]$, satisfies:

$$\begin{cases} u_{ij} \in \{0, 1\}, & \forall i, j \\ 0 < \sum_{j=1}^N u_{ij} < N & \forall i \\ \sum_{i=1}^C u_{ij} = 1 & \forall j \end{cases} \quad (1)$$

For the fuzzy case, \mathbf{U} satisfies [43]:

$$\begin{cases} u_{ij} \in [0, 1], & \forall i, j \\ 0 < \sum_{j=1}^N u_{ij} < N & \forall i \\ \sum_{i=1}^C u_{ij} = 1 & \forall j \end{cases} \quad (2)$$

1 K-Means Algorithm

In [41], the author describes the K-Means algorithm, one of the earliest and simplest unsupervised algorithms for solving the well known clustering problem, partitioning N feature vectors into C clusters. The process is based on iteratively minimizing an objective function known as the vector quantization error, or distortion. In particular, it minimizes

$$J(\mathbf{B}, \mathbf{U}; \mathcal{X}) = \sum_{i=1}^C \sum_{j=1}^N (u_{ij}) d(\mathbf{x}_j, \beta_i). \quad (3)$$

subject to the constraints in (1). One setback to this method is that the minimization of (3) has no closed form solution. Therefore, a local minimization of J could be achieved by alternating optimization. The first step of the algorithm fixes the cluster parameters β_i , or the cluster centers, and assigns points to the nearest cluster based on a given distance $d(\mathbf{x}_j, \beta_i)$. Typically, the Euclidean distance

$$d(\mathbf{x}_j, \beta_i) = d_{ij} = \|\mathbf{x}_j - \mathbf{c}_i\|^2 \quad (4)$$

is used in the k-means algorithm. The next step fixes the memberships, u_{ij} , and J is optimized with respect to the clusters' centroids. This yields an update equation for the cluster centers:

$$\mathbf{c}_i = \frac{\sum_{j=1}^N u_{ij} \mathbf{x}_j}{\sum_{j=1}^N u_{ij}}. \quad (5)$$

The resulting k-means algorithm is outlined below:

K-Means Algorithm

Fix the number of clusters C ;
Initialize the cluster centroids;
Repeat
 Assign each point x_i to the nearest cluster
 Update the centroids c_j using (5);
Until(centers stabilize)

2 The Fuzzy C-Means (FCM) Algorithm

The FCM algorithm is a modification of the K-Means algorithm that changes the assignment paradigm from crisp to fuzzy [40]. The algorithm minimizes:

$$J(\mathbf{B}, \mathbf{U}; \mathcal{X}) = \sum_{i=1}^C \sum_{j=1}^N (u_{ij})^m d(\mathbf{x}_j, \beta_i), \quad (6)$$

subject to the constraints in (2). In (6), $m \in (1, \infty)$ is a weighting exponent (called the fuzzifier) and $d(\mathbf{x}_j, \beta_i)$ is the distance from feature point \mathbf{x}_j to prototype β_i . Minimization of (6) with respect to \mathbf{U} , subject to the constraints in (2), yields [40]

$$\left. \begin{aligned} u_{ij} &= \frac{1}{\sum_{k=1}^C \left(\frac{d(\mathbf{x}_j, \beta_i)}{d(\mathbf{x}_j, \beta_k)} \right)^{\frac{1}{m-1}}} && \text{if } I_j = 0 \\ u_{ij} &= 0 && \text{if } i \notin I_j \\ \sum_{i \in I_j} u_{ij} &= 1 && \text{if } i \in I_j \end{aligned} \right\} \text{ if } I_j \neq 0 \quad (7)$$

where $I_j = \{i | 1 \leq i \leq C, d(\mathbf{x}_j, \beta_i) = 0\}$.

Minimization of (6) with respect to the prototype parameters β is dependent on the distance measure. In the initial formulation, the Euclidean distance given by (4) was used. This distance allows the FCM to find spherical clusters. In this case the prototypes, β , are the clusters' centers. The update equation of the centroids is obtained by fixing the membership values and minimizing (6) with respect to \mathbf{c}_i . This minimization yields

$$\mathbf{c}_i = \frac{\sum_{j=1}^N (u_{ij})^m \mathbf{x}_j}{\sum_{j=1}^N (u_{ij})^m}. \quad (8)$$

The FCM algorithm is summarized below:

Fuzzy C-Means Algorithm

```

Fix the number of clusters  $C$ ;
Fix  $m$ ,  $m \in (1, \infty)$ ;
Initialize the cluster centroids;
Repeat
    Compute  $d(\mathbf{x}_j, \beta_i)$  using (4);
    Update the partition matrix  $U^{(k)}$  using (7);
    Update the centers using (8);
Until(centers stabilize)

```

3 The Gustafson-Kessel (GK) Algorithm

All prototype-based clustering algorithms require the use of some form of similarity measure, typically calculated using a distance measure. The K-Means and FCM algorithms utilize the Euclidean distance in (4) which provides the means for obtaining spherical shaped clusters. However, in many applications, clusters, even within the same data set, can have different geometric shapes. In [44], Gustafson and Kessel proposed modifying the FCM algorithm to identify clusters with various shapes. Instead of the Euclidean distance, the authors use an A-Norm distance given by:

$$d(\mathbf{x}_j, \beta_i) = \| \mathbf{x}_j - c_i \|_{\mathbf{A}_i}^2 = (\mathbf{x}_j - c_i) \mathbf{A}_i (\mathbf{x}_j - c_i), \quad (9)$$

subject to

$$\det(\mathbf{A}_i) = \rho_i \text{ (constant)} \quad \forall i. \quad (10)$$

Fixing the determinant and varying \mathbf{A} allows the algorithm to search for a cluster shape that fits the data while preserving the *volume* of the cluster. Each cluster i , is represented by an independent matrix \mathbf{A}_i . Thus, the GK algorithm is able to find ellipsoidal clusters of different sizes and orientations.

The objective function for the GK algorithm is the same as the FCM (6), and minimization yields the same equations for updating the centers (8) and the memberships (7). For optimizing the objective function with respect to (A), the

authors show that for each matrix \mathbf{A}_i , \mathbf{A}_i^* is a local minimum of J if

$$\mathbf{A}_i^* = [\rho_i |\mathbf{C}_i|]^{1/2} \mathbf{C}_i^{-1} \text{ for } 1 \leq i \leq C,$$

where

$$\mathbf{C}_i = \frac{\sum_{j=1}^N (u_{ij})^m (\mathbf{x}_j - \mathbf{c}_i)(\mathbf{x}_j - \mathbf{c}_i)^T}{\sum_{j=1}^N (u_{ij})^m} \quad (11)$$

is the fuzzy covariance matrix. Using this matrix, and $\rho_i = 1$, the distance in equation (9) reduces to

$$d(\mathbf{x}_j, \beta_i) = |\mathbf{C}_i|^{1/n} (\mathbf{x}_j - \mathbf{c}_j)^T \mathbf{C}_i^{-1} (\mathbf{x}_j - \mathbf{c}_i). \quad (12)$$

The GK algorithm is summarized below:

GK Algorithm

Fix the number of clusters C ;
 Fix m , $m \in (1, \infty)$;
 Initialize the cluster centroids;
Repeat
 Compute $d(\mathbf{x}_j, \beta_i)$ using (12);
 Update the partition matrix $U^{(k)}$ using (7);
 Update the centers using (8);
 Update the covariance matrix using (11);
Until(centers stabilize)

B Determining the Optimal Number of Clusters

One of the primary impediments of most clustering algorithms is the necessity for the number of clusters C to be specified and set prior to beginning the clustering process. In reality, the selection of this parameter is not only critical to the algorithm's success, but information leading to a reliable value may not be available. The high dimensionality data [45] and the sparsity of the search space [46] makes this problem more acute.

Several approaches have been proposed to find the optimal number of clusters, C , [47, 43, 48, 49, 50, 51, 52, 42, 53, 54, 55]. In general, these methods can be categorized into validity-based approaches, and objective function based approaches.

1 Validity-Based Approaches

An intuitive method for determining the optimal number of clusters in a given data set is to use cluster validity measures [47, 43]. In general, these measures are an iterative attempt to maximize the densities of the clusters. In [49], a predetermined upper limit for the number of clusters is defined, C_{max} , then the data is clustered while varying the number of clusters from 2 to C_{max} . Two values are computed based on the cluster layout and are used to assess the validity of the clustering results. The first one is the intra-cluster distance and can be used as an indication of the compactness of the clusters. It is defined as

$$D_{intra} = \frac{1}{N} \sum_{i=1}^C \sum_{x \in C_i} \|x - c_i\|^2. \quad (13)$$

In (13), N is the size of the data set, C is the current number of clusters, and c_i is the centroid of cluster i . The second measure is the inter-cluster distance and can be used as an indication of the separability of the clusters. It is defined as

$$D_{inter} = \min(\|c_i - c_j\|^2), \quad i = 1, 2, \dots, C-1 \\ j = i+1, \dots, C \quad (14)$$

The inter- and intra-cluster measures are then combined to form an overall validity measurement defined as:

$$validity = \frac{D_{intra}}{D_{inter}}. \quad (15)$$

Using this validity measure, the value of C that minimizes (15) can be selected as the optimal number of clusters.

In [48], the authors propose the average partition density and the hypervolume to evaluate the compactness of the clusters. Particularly, the hypervolume, V , is defined as

$$V = \sum_{i=1}^C [\det \mathbf{C}_i]^{\frac{1}{2}} \quad (16)$$

where \mathbf{C}_i is the fuzzy covariance matrix of cluster i (as defined in (11)) with $m = 1$.

The average paritional density, D_{PA} , is defined by:

$$D_{PA} = \frac{1}{C} \sum_{i=1}^C \frac{S_i}{[\det \mathbf{C}_i]^{\frac{1}{2}}}. \quad (17)$$

In (17) S_i is the sum of central members given by:

$$S_i = \sum_{k \in A} u_{ik}, \quad (18)$$

where

$$A = \{j | (\mathbf{x}_j - c_i)^T \mathbf{C}_i^{-1} (\mathbf{x}_j - c_i) < 1\}. \quad (19)$$

In other words, S_i accounts for core objects with a Mahalanobis distance less than one. This validity measure was modified in [55, 56] in order to measure the validity of spherical, elliptic, and quadratic shell clusters.

2 Objective Function Based Approach

The Competitive Agglomeration (CA) [42] algorithm provides an efficient clustering method incorporating both partitional and hierarchical clustering to automatically determine the best C by minimizing:

$$J(\mathbf{B}, \mathbf{U}, \mathcal{X}) = \sum_{i=1}^C \sum_{j=1}^N (u_{ij})^2 d(\mathbf{x}_j, \beta_i) - \alpha \sum_{i=1}^C \left[\sum_{j=1}^N u_{ij} \right]^2, \quad (20)$$

subject to the constraints in (2). In (20), $d(\mathbf{x}_j, \beta_i)$ represents the distance from feature vector \mathbf{x}_j to prototype β_i . The number of clusters, C , is dynamically updated. Optimization of J with respect to \mathbf{U} yields [42]:

$$u_{st} = u_{st}^{\text{FCM}} + u_{st}^{\text{BIAS}}, \quad (21)$$

where

$$u_{st}^{\text{FCM}} = \frac{[1/d(\mathbf{x}_t, \beta_s)]}{\sum_{k=1}^C [1/d((x)_t, \beta_k)]}, \quad (22)$$

and

$$u_{st}^{\text{BIAS}} = \frac{\alpha}{d(\mathbf{x}_t, \beta_s)} (N_s - \bar{N}_t). \quad (23)$$

In (23)

$$N_s = \sum_{j=1}^N u_{sj} \quad (24)$$

is the fuzzy cardinality of cluster s , and

$$\bar{N}_t = \frac{\sum_{k=1}^C [1/d(\mathbf{x}_t, \beta_k)] N_k}{\sum_{k=1}^C [1/d(\mathbf{x}_t, \beta_k)]}. \quad (25)$$

The constant α term in (20) is used to balance the two terms of the objective function. Therefore, it can be seen as a weighting term measuring the importance of the regularization term with respect to the sum of intra-cluster distances. In [42], the authors recommend estimating α in every iteration k using

$$\alpha(k) = \eta_0 \exp(-k/\tau) \frac{\sum_{i=1}^C \sum_{j=1}^N (u_{ij})^2 d(\mathbf{x}_j, \beta_i)}{\sum_{i=1}^C [\sum_{j=1}^N u_{ij}]^2} \quad (26)$$

where η_0 is the initial value, and τ the time constant.

The CA algorithm is summarized below:

Competitive Agglomeration Algorithm

Fix the maximum number of clusters $C = C_{\max}$;
Initialize iteration counter $k = 0$ and the fuzzy C partition $\mathbf{U}^{(0)}$;
Compute initial cardinalities N_i for $1 \leq i \leq C$ using (24);
Repeat
 Compute $d(\mathbf{x}_j, \beta_i)$ for $1 \leq i \leq C$ and $1 \leq j \leq N$;
 Update $\alpha(k)$ using (26);
 Update the partition matrix $\mathbf{U}^{(k)}$ using (21);
 Compute the cardinality N_i for $1 \leq i \leq C$ using (24);
 If $(N_i < \epsilon_1)$ discard cluster β_i ;
 Update the number of clusters C ;
 Update the prototype parameters;
 $k = k + 1$;
Until(prototype parameters stabilize)

C Feature Selection and Weighing

Another hindrance in creating a good learning algorithm is the selection of feature subsets that best represent the overall data. In fact, the use of irrelevant features can severely degrade the performance of the clustering algorithm. Using supervised learning approaches, several methods have been proposed to perform feature selection and weighing. Feature selection methods determine which features are relevant and discard the rest, where feature weighing methods assign continuous weights to all features based on their relevance. These methods are developed using a variety of schema, such as genetic algorithms [57, 58], supervised fuzzy clustering [59], feature correlation [60], feature similarity [61], cross-validation [62], feature space reduction [63, 64, 65, 66], and other novel methods [67, 68, 69, 70, 71, 72, 73].

Although the area of feature selection and weighing has a wide field of study in supervised clustering, the domain of unsupervised clustering has not shared the same interest level and only few methods have been proposed [65, 66, 69]. In the following, we outline the Simultaneous Clustering and Attribute Discrimination (SCAD) algorithm [74, 75, 76] since it is highly relevant to the proposed approach.

In [74, 75], the authors proposed an algorithm that performs Simultaneous Clustering and Attribute Discrimination (SCAD). It minimizes

$$J(\mathbf{B}, \mathbf{U}, \mathbf{V}; \mathcal{X}) = \sum_{i=1}^C \sum_{j=1}^N (u_{ij})^m \sum_{k=1}^n (v_{ik})^q d_{ijk}^2 \quad (27)$$

subject to the membership constraint in (2), and

$$v_{ik} \in [0, 1] \forall i, k; \quad \text{and} \quad \sum_{k=1}^n v_{ik} = 1, \forall i. \quad (28)$$

In [75], $q \in (1, \infty)$ is referred to as the discrimination exponent.

Minimization of J with respect to \mathbf{V} yields

$$v_{ik} = \frac{1}{\sum_{t=1}^n \left(\tilde{D}_{ik} / \tilde{D}_{it} \right)^{1/(q-1)}}, \quad (29)$$

where $\tilde{D}_{ik} = \sum_{j=1}^N (u_{ij})^m d_{ijk}^2$ is the measure of dispersion of the i^{th} cluster along the k^{th} dimension, and $\sum_{t=1}^n \tilde{D}_{it}$ is the total dispersion of the i^{th} cluster. In other words, the more compact the i^{th} cluster is along the k^{th} dimension (smaller \tilde{D}_{ik}), the higher the relevance weight, v_{ik} will be for the k^{th} feature.

Minimization of J with respect to \mathbf{U} subject to the constraints in (2) yields

$$u_{ij} = \frac{1}{\sum_{k=1}^C \left[\tilde{d}_{ij}^2 / \tilde{d}_{kj}^2 \right]^{\frac{1}{m-1}}}. \quad (30)$$

where

$$\tilde{d}_{ij}^2 = \sum_{k=1}^n v_{ik} d_{ijk}^2. \quad (31)$$

Minimization of J with respect to the centers \mathbf{C} yields:

$$c_{ik} = \begin{cases} 0 & \text{if } v_{ik} = 0, \\ \frac{\sum_{j=1}^N (u_{ij})^m x_{jk}}{\sum_{j=1}^N (u_{ij})^m} & \text{if } v_{ik} > 0. \end{cases} \quad (32)$$

The SCAD algorithm is summarized below:

**Simultaneous Clustering and
Attribute Discrimination Algorithm**

Fix the number of clusters C ;
Fix m , $m \in (1, \infty)$;
Fix the discrimination exponent q , $q \in (1, \infty)$;
Initialize the centers and fuzzy partition matrix \mathbf{U} ;
Initialize all the relevance weights to $1/n$;
Repeat
 Compute d_{ijk}^2 for $1 \leq i \leq C$, $1 \leq j \leq N$, and $1 \leq k \leq n$;
 Update the relevance weights matrix \mathbf{V} by using equation (29);
 Compute \tilde{d}_{ij}^2 by using equation (31);
 Update the partition matrix U by using equation (30);
 Update the centers by using equation (32);
Until(centers stabilize)

The SCAD algorithm is designed to perform clustering and feature weighting simultaneously while searching for optimal cluster parameters and relevance weights. In the case of high dimensional data, learning weights for each feature could lead to over-fitting. In [77], a coarse approach to feature weighting (SCAD_c) was proposed to avoid this problem. In SCAD_c, the features are divided into logical subsets and a relevance weight is learned for each subset as opposed to learning a feature weight for each feature. The authors in [77] also introduce the SCADCA algorithm, which combines the benefits of feature weighting from the SCAD_c algorithm, with the search for an unknown number of clusters from the CA algorithm [42].

D Semi-Supervised Clustering

Unsupervised methods are used for organization and classification purposes due to the lack of labels for the provided data. In most applications, clustering can be a challenging task. These algorithms tend to get trapped in local minima due to complex objective functions. Moreover, unsupervised clustering methods form clusters based solely on the similarity between objects provided by a given similarity measure. Therefore, these methods rely heavily on the choice of similarity measure and may not provide semantically meaningful clusters. Alternatively, the use of supervised methods may not be an option because labeling all the data could be a very expensive or even an impossible task. For instance, in large data sets the overall number of classes may not even be known.

Recent research into an area known as semi-supervised clustering has been proposed in an attempt to improve the performance of unsupervised learning [78, 27, 25, 26, 79]. Semi-supervised methods are formulated using information garnished from side information associated with the data in order to *guide* or *adjust* the clustering process. Typically, the information is presented in the form of labels [80, 81, 78],

constraints [82, 83, 27], or hints [84]. In real world applications, the true labels of the data may not be known. Therefore, it may be more practical to specify which pairs of points should or should not belong to the same cluster. Thus, it is more practical to incorporate the partial supervision in the form of constraints as opposed to class labels. Although semi-supervised methods have been proven to outperform their unsupervised counterparts, the extensiveness of the research into semi-supervised methods is not as vast [85, 86, 26].

The majority of semi-supervised clustering methods can be dichotomized as *similarity-adapting* [83, 87, 88, 89, 90] or *search-based* [28, 82, 25, 78, 27, 26]. Since our approach uses a *search-based* method of semi-supervision, only *search-based* methods will be outlined in this literature survey.

Search-based semi-supervised clustering methods adapt existing clustering algorithms, using constraints or labels, to bias the search for a semantically more meaningful partition, and to guide the algorithm to reach the global optimum. There are various methods in which this information can be incorporated. Some methods use constraints to perform transitive closure and initialize the clustering process [78]. Other methods incorporate the constraints into the optimization process. In the latter approach, some methods require that all constraints be satisfied during the clustering process [28], while some other methods use modifications to the objective function, penalizing the process when constraints are left unsatisfied [82].

One method of incorporating the background knowledge is to use pairwise constraints [25, 83, 91, 26]. Generally, these constraints are defined as either *must-link* or *cannot-link* [82]. When two points are required to be in the same cluster they form a *must-link* constraint. In contrast, if the two points are intended to be in different clusters they form a *cannot-link* constraint. These constraints are formulated in order to guide the clustering process to find both naturally occurring patterns with

a user-defined overtone. The constraint satisfaction criteria can be defined as either being strict or relaxed. If the satisfaction criterion is defined as strict the constraints are to be unconditionally satisfied [28, 25], where a relaxed criterion implies that the constraints may or may not be satisfied [92].

The following sections describe some clustering algorithms that incorporate pairwise constraints, including those that use *must* and *cannot* link constraints, and implement strict and relaxed satisfaction criterion.

1 The K-Means Algorithm with Pairwise Constraints

In [25] and [91] the authors adapted the K-Means algorithm [41] to incorporate pairwise constraints. In [25], Wagstaff et al. purposed an algorithm that uses a strict satisfaction criteria with must- and cannot-link constraints. Davidson et al. in [91] implemented must- and cannot-link constraints along with minimal separation and cluster density constraints in conjunction with a relaxed satisfaction methodology.

Another adaptation of the K-Means algorithm was proposed in [82]. In this approach, a constrained k-means algorithm (COP-KMeans) was proposed by modifying the cluster assignment process, while leaving the optimization process unaltered. During the cluster assignment step of COP-KMeans, each point x_j is checked against the set of constraints in order to ensure that there are no violations. The algorithm attempts to assign each point x_j to the nearest cluster i . If a violation exists, then the algorithm parses each remaining cluster to find the closest cluster in which no violation occurs.

The COP-KMEANS algorithm is summarized below:

COP-KMEANS Algorithm

```

Fix the number of clusters  $C$ ;
Randomly initialize the centroids;
Repeat
    Assign each point  $s_j$  to the nearest cluster using
    the Euclidean distance and
    VIOLATE-CONSTRAINT = false;
    Update the centroids  $c_i$  using (5);
Until(centers stabilize)

```

Davidson et al. [91] implemented another modification to the k-means algorithm by adapting the center and error updating functions, while introducing two other forms of constraints. The authors introduce δ - and ϵ -constraints which act upon *groups* of instances. The δ -constraints, or minimum separation constraints, require the distance between constraints to be at least δ , or for any two points x_i and x_j in different clusters the distance $d(x_i, x_j) > \delta$. The ϵ -constraints require that any two points x_i and x_j contained in the same cluster have a distance $d(x_i, x_j) \leq \epsilon$ ensuring that the formulated clusters are dense. The δ - and ϵ -constraints can also be written as a conjunction and disjunction of must-link constraints respectively. Therefore, all four types of constraints in this method can be broken down into the traditional must- and cannot-link constraints.

Let c_j be the centroid of the j^{th} cluster, Q_j be the set of points that are closest to the j^{th} centroid, M is a collection of must-link constraints, and C is a collection of cannot-link constraints. The constrained k-means algorithm minimizes:

$$CVQE_j = \frac{1}{2} \sum_{s_i \in Q_j} T_{j,1} + \frac{1}{2} \sum_{l=1, g(l)=j}^{s+r} (T_{j,2} \times T_{j,3}) \quad (33)$$

where

$$\begin{aligned}
T_{j,1} &= (C_j - s_i)^2, \\
T_{j,2} &= [(C_j - C_{g'(l)})^2 \neg \Delta(g'(l), g(l))]^{ml}, \\
T_{j,3} &= [(C_j - C_{h(g'(l))})^2 \Delta(g(l), g'(l))]^{1-ml}.
\end{aligned} \tag{34}$$

In (34), $g(i)$ and $g'(i)$ return the cluster index of the 1st and 2nd instances of the i^{th} constraint. The subscript index, $h(i)$ returns the next index of the cluster whose centroid is closest to the i^{th} cluster centroid. The function Δ is defined such as $\Delta(x, y) = 1$ if $x = y$ and 0 otherwise, while $\neg \Delta$ is the negation of the Δ function. The superscript, ml , indicates that the current pair of objects is a must-link ($ml = 1$) or a cannot-link ($ml = 0$). In (33), $T_{j,1}$ is the original K-means error function, and $T_{j,2}$ is the cost of violating a must-link constraint, while $T_{j,3}$ is the cost of violating a cannot-link constraint.

Minimizing the objective function in (33), it can be shown [91] that the new centroid update function is:

$$C_j = \frac{Y_j}{Z_j} \tag{35}$$

where

$$Y_j = \sum_{s_i \in Q_j} s_i + \sum_{l=1, g(l)=j, \Delta(g(l), g'(l))=0}^s C_{g'(l)} + \sum_{l=s+1, g(l)=j, \Delta(g(l), g'(l))=1}^{s+r} C_{h(g'(l))}$$

and

$$Z_j = |Q_j| + \sum_{g(l)=j, l=1}^s (1 - \Delta(g(l), g'(l))) + \sum_{g(l)=j, l=s+1}^{s+r} \Delta(g(l), g'(l))$$

Intuitively, the centroid update function moves the cluster centroid of a violated must-link constraint closer to the cluster containing the other point of the pair so that one of the points will shift to the correct cluster. Similarly the update rule for a cannot-link violation moves the centroid containing both constrained instances to the nearest cluster centroid so that one of the instances may be assigned to it, satisfying the constraint. All constraints in this system were generated randomly.

The Constrained K-means algorithm is summarized below

Constrained K-Means Algorithm

Fix the number of clusters C ;
 Randomly initialize the centroids;
Repeat
 Assign each point s_i to the nearest cluster using
 the Euclidean distance;
 Compute $CVQE$ using (33);
 Update the centroids C_j using (35);
Until(centers stabilize)

2 Pairwise-Constrained Competitive Agglomeration

The authors in [26] proposed a semi-supervised clustering algorithm that uses Pairwise-Constraints and Competitive Agglomeration (PCCA). The PCCA algorithm combines features from the CA algorithm [42] with features from previous work on semi-supervised clustering [28, 82, 78]. Using the same notation as in the CA, the PCCA algorithm minimizes the following objective function:

$$\begin{aligned}
 J(\mathbf{B}, \mathbf{U}, \mathcal{X}) = & \sum_{k=1}^C \sum_{i=1}^N (u_{ik})^2 d(x_i, \beta_k) \\
 & + \gamma \left(\sum_{(x_i, x_j) \in \mathcal{M}} \sum_{k=1}^C \sum_{l=1, l \neq k}^C u_{ik} u_{jl} + \sum_{(x_i, x_j) \in \mathcal{C}} \sum_{k=1}^C u_{ik} u_{jk} \right) \\
 & - \alpha \sum_{k=1}^C \left[\sum_{i=1}^N u_{ik} \right]^2
 \end{aligned} \tag{36}$$

subject to the constraints in (2). In (36), \mathcal{M} is the set of available must-link constraints, i.e. $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}$ implies that \mathbf{x}_i and \mathbf{x}_j must be assigned to the same cluster, and \mathcal{C} the set of cannot-link constraints, i.e. $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}$ implies that \mathbf{x}_i and \mathbf{x}_j cannot be assigned to the same cluster.

The first term in (36) is the sum of squared distances to the prototypes weighted by the membership and is from the FCM objective function [40]. This term is used to seek compact clusters. The second term is the costs of violating the pairwise must- and cannot-link constraints. The penalty for two points in different clusters (for

must-link constraints) or in a same cluster (for cannot-link constraints) is weighted by their membership values. The third term is the sum of squares of the cardinality of the clusters (from the CA objective function) and controls the competition between clusters. The second term in (36) is weighted by γ , a constant factor that controls the importance of the supervision. In, [79], the authors recommend the estimation of γ using

$$\gamma = \frac{N \sum_{k=1}^C \sum_{i=1}^N u_{ik}^2 d(\mathbf{x}_i, \beta_k)}{M \sum_{k=1}^C \sum_{i=1}^N u_{ik}^2}, \quad (37)$$

where M is the number of pairwise constraints.

The value of α in (36) controls the competition between clusters. In [26] the authors recommend that the value of α be updated in every iteration using

$$\begin{aligned} \alpha(k) = \eta_0 \exp(-k/\tau) & \frac{\sum_{i=1}^C \sum_{j=1}^N (u_{ij})^2 d(\mathbf{x}_j, \beta_i)}{\sum_{i=1}^C [\sum_{j=1}^N u_{ij}]^2} \left[\sum_{j=1}^C \sum_{i=1}^N (u_{ij})^2 d(\mathbf{x}_i, \beta_j) \right. \\ & \left. + \gamma \left(\sum_{(x_i, x_j) \in \mathbf{M}} \sum_{k=1}^C \sum_{l=1, l \neq k}^C u_{ik} u_{jl} + \sum_{(x_i, x_j) \in \mathbf{C}} \sum_{k=1}^C u_{ik} u_{jk} \right) \right], \end{aligned} \quad (38)$$

where η_0 is the initial value, and τ is the time constant. When all terms are combined and an appropriate α has been selected, the final partition of the data will minimize the sum of the intra-cluster distances, while creating the smallest number of clusters that satisfies the given constraints as much as possible.

Minimizing J with respect to \mathbf{U} , subject to the constraints in (2) yields [26]

$$u_{rs} = u_{rs}^{\text{FCM}} + u_{rs}^{\text{Constraints}} + u_{rs}^{\text{Bias}} \quad (39)$$

where

$$u_{rs}^{\text{FCM}} = \frac{\frac{1}{d(\mathbf{x}_r, \beta_s)}}{\sum_{k=1}^C \frac{1}{d(\mathbf{x}_r, \beta_k)}}, \quad (40)$$

$$u_{rs}^{\text{Constraints}} = \frac{\gamma}{2d(\mathbf{x}_r, \beta_s)} (\overline{C}_{v_r} - C_{v_{rs}}), \quad (41)$$

and

$$u_{rs}^{\text{Bias}} = \frac{\alpha}{d(\mathbf{x}_r, \beta_s)} (N_s - \overline{N}_r). \quad (42)$$

In (41), $C_{v_{rs}}$ and \bar{C}_r are defined as

$$C_{v_{rs}} = \sum_{(\mathbf{x}_r, \mathbf{x}_j) \in \mathcal{M}} \sum_{l=1, l \neq s}^C u_{jl} + \sum_{(\mathbf{x}_r, \mathbf{x}_j) \in \mathcal{C}} u_{js}, \quad (43)$$

and

$$\bar{C}_{v_r} = \frac{\sum_{k=1}^C \frac{\left(\sum_{(\mathbf{x}_r, \mathbf{x}_j) \in \mathcal{M}} \sum_{l=1, l \neq k}^C u_{jl} + \sum_{(\mathbf{x}_r, \mathbf{x}_j) \in \mathcal{C}} u_{jk} \right)}{d(\mathbf{x}_r, \beta_k)}}{\sum_{k=1}^C \frac{1}{d(\mathbf{x}_r, \beta_k)}}, \quad (44)$$

In (42), \bar{N}_r is defined as

$$\bar{N}_r = \frac{\sum_{k=1}^C \frac{N_k}{d(\mathbf{x}_r, \beta_k)}}{\sum_{k=1}^C \frac{1}{d(\mathbf{x}_r, \beta_k)}} \quad (45)$$

The first term in equation (39), u_{rs}^{FCM} , is the same as the FCM membership equation and considers only the relative distances between data items and prototypes. The second term, $u_{rs}^{\text{Constraints}}$, takes into account the available supervision. Memberships are depreciated or reinforced depending on the satisfaction of the pairwise constraints. The third term, u_{rs}^{Bias} , leads to a reduction of the cardinality of spurious clusters, which are discarded when their cardinality drops below a threshold.

The PCCA algorithm was formulated using the GK distance. Thus, each prototype consists of a center and a covariance matrix. Since the second and third terms in (36) do not depend explicitly on the prototype parameters, it can be easily shown that these parameters are updated as in the GK algorithm. That is the centers are updated using (8) and the covariance matrix is updated using (11).

The PCCA algorithm is summarized below:

The Pairwise-Constrained Competitive Agglomeration

Fix the number of clusters C ;
Randomly initialize prototypes $u_j, j \in \{1, \dots, C\}$;
Initialize memberships u_{ij} : equal membership of every
data item to every cluster;
Compute initial cardinalities N_j using (24);
Repeat
 Update α and γ using (38) and (37);
 Update the memberships u_{ij} using (39);
 Update the cardinalities $N_j, j \in \{1, \dots, C\}$, using (24);
 For $j \in \{1, \dots, C\}$, if $N_j < \text{threshold}$ then discard cluster j ;
 Update the number of clusters C ;
 Update the centers and covariance matrix using
 (8) and (11) respectively.
Until(prototypes stabilize)

We should note here that if the number of clusters is fixed, and binary memberships are used, the PCCA algorithm reduces to the PCKmeans algorithm given in [78].

CHAPTER III

ADAPTIVE CONSTRAINED CLUSTERING

This chapter presents our proposed approach to clustering called Adaptive Constrained Clustering (ACC). This algorithm combines the benefits of the cluster reduction techniques of the CA algorithm [42] (refer to Section §II.B.2), with the ability to learn cluster-dependent feature relevance weights from the SCAD algorithm [74, 75] (§II.C), and the incorporation of partial supervision from the PCCA algorithm [26] (refer to §II.D.2). The proposed algorithm uses a single objective function formulated to jointly optimize all of the above criteria. We derive the necessary conditions to optimize the joint objective function and describe the different steps involved in the algorithm. We show that, the ACC algorithm can be used to adaptively cluster a given data set using partial supervision information to guide the clustering process, learn cluster-dependent feature relevance weights, and find the optimal number of clusters.

A The Constrained Clustering (CC) Algorithm

Let $\mathbf{X} = \{x_j \in \mathbb{R}^p | j = 1, \dots, N\}$ be a set of N feature vectors in an p -dimensional feature space. Let $\mathbf{B} = (\beta_1, \dots, \beta_c)$ represent a C -tuple of prototypes each of which characterizes one of the C clusters. Each β_i consists of a set of parameters. Let u_{ij} represent the grade of membership of feature point x_j in cluster β_i . Our

approach is fuzzy and thus, u_{ij} , satisfies the following membership constraint:

$$\begin{cases} u_{ij} \in [0, 1], & \forall i, j \\ 0 < \sum_{j=1}^N u_{ij} < N & \forall i \\ \sum_{i=1}^C u_{ij} = 1 & \forall j \end{cases} \quad (46)$$

As in SCAD_c [77], we assume that the p features are partitioned into K logical subsets: FS_1, FS_2, \dots, FS_K , and that each subset, FS_s , includes k_s features. This partitioning is application dependent. For example, in imaging applications, these subsets could be formed by separating color features into one subset, texture features into a second subset, and another subset for structure features. Let d_{ijs} be the partial distance between feature \mathbf{x}_j and cluster i using the s^{th} feature subset. The distances used for each subset are independent, and it is not necessary that these distances be the same. In the previous example, it is possible that similarity for the color feature subset be characterized using the Euclidean distance, while the texture features and the structure features could be represented by the Mahalanobis and L_p norm distances respectively. The only requirement is that the different distance measures yield values within the same dynamic range. Then, the total distance, D_{ij} , between \mathbf{x}_j and cluster i is computed using a simple weighted average operator to aggregate the partial degrees of similarity and their weights. That is, we let

$$D_{ij} = \sum_{s=1}^K v_{is} d_{ijs}. \quad (47)$$

In (47), v_{is} is the relevance weight of feature subset FS_s , with respect to cluster i and satisfies the following constraints:

$$v_{ik} \in [0, 1] \forall i, k; \quad \text{and} \quad \sum_{k=1}^n v_{ik} = 1, \forall i. \quad (48)$$

In contrast to the definitions in [25, 26, 91] related to *must* and *cannot* link constraints, the constraints used in our approach are soft and are defined as *should* and *should-not* link constraints. The original constraint definitions (e.g. in [25, 79]) imply

that the satisfaction criteria is hard, that is, the pairs of instances contained in *must* and *cannot* link constraints need to be satisfied unconditionally regardless of the clusters' distributions. The proposed CC algorithm does not employ a hard satisfaction criteria. Therefore, the use of the must- and cannot-link can be semantically misleading. Since the use of constraints are merely suggestions based on previous knowledge on how the clusters *should* be formed, the algorithm then uses these constraints to guide the clustering process but does not guarantee their satisfaction.

Let \mathcal{S} be the set of available *should* link constraints, i.e. $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}$ implies that \mathbf{x}_i and \mathbf{x}_j should be assigned to the same cluster. Similarly, let \mathcal{N} the set of should not-link constraints, i.e. $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{N}$ implies that \mathbf{x}_i and \mathbf{x}_j should be assigned to different clusters.

The Constrained Clustering (CC) algorithm minimizes the following objective function:

$$\begin{aligned}
J(\mathbf{B}, \mathbf{U}, \mathbf{V}; \mathcal{X}) = & \sum_{i=1}^C \sum_{j=1}^N u_{ij}^m \sum_{s=1}^K v_{is}^q d_{ijs} \\
& + \gamma \left(\sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \mathcal{S}} \sum_{i=1}^C \sum_{l=1, l \neq i}^C \kappa_{jk} u_{ji}^m u_{kl}^m + \sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \mathcal{N}} \sum_{i=1}^C \rho_{jk} u_{ji}^m u_{ki}^m \right) \quad (49) \\
& - \alpha \sum_{i=1}^C \left[\sum_{j=1}^N u_{ij} \right]^2.
\end{aligned}$$

subject to the constraints in (46), and (48). The first term in (49) is the objective function of the SCAD algorithm [75], and is used to search for compact clusters with their feature relevance weights. The second term is composed of the cost of violating the pairwise *should* link and *should-not* link constraints. The penalty terms are weighted by the membership values of the points that violate the constraints. In other words, the penalty term is greater when the points are part of the core of the cluster (high membership), than if the points were on the border of the cluster (low membership). Unlike previous work on semi-supervised clustering, our approach

does not treat each constraint equally important. To that effect, κ_{jk} , is an application dependent penalty weight for violating a *should* link constraint and ρ_{jk} is the penalty weight for violating a *should-not* link constraint between \mathbf{x}_i and \mathbf{x}_j .

The third term in (49) is the sum of the squared cardinalities. This is a regularization term that is used to introduce competition among the clusters and promote sparsity. It allows the algorithm to partition the data into the optimal number of clusters. In (49), γ is a constant weighing the importance of the supervision compared to the sum of intra-cluster distances. Similarly, α is a constant that controls the agglomeration rate. With the proper selection of these constraints and the combination of the three terms, the CC algorithm will seek the optimal number of clusters, their parameters and the feature relevance weights while minimizing the sum of intra-cluster distances and the number of violated constraints.

To optimize of J with respect to the membership U , we apply the Lagrange multiplier technique and obtain

$$\begin{aligned}
J(\mathbf{V}, \mathbf{\Lambda}) = & \sum_{i=1}^C \sum_{j=1}^N u_{ij}^m \sum_{s=1}^K v_{is}^q d_{ijs} \\
& + \gamma \left(\sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \mathcal{S}} \sum_{i=1}^C \sum_{l=1, l \neq i}^C \kappa_{jk} u_{ji}^m u_{kl}^m + \sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \mathcal{N}} \sum_{i=1}^C \rho_{jk} u_{ji}^m u_{ki}^m \right) \\
& - \alpha \sum_{i=1}^C \left[\sum_{j=1}^N u_{ij} \right]^2 \\
& - \sum_{j=1}^N \lambda_j \left(\sum_{i=1}^C u_{ji} - 1 \right).
\end{aligned} \tag{50}$$

In (50), $\mathbf{\Lambda} = [\lambda_1, \dots, \lambda_c]^t$ is a vector of Lagrange multipliers corresponding to the C constraints in (46). The necessary conditions for updating the memberships, u_{ij} , can

be obtained by fixing the parameters \mathbf{B} and \mathbf{V} , and solving $\frac{\partial J}{\partial u_{ij}} = 0$. Doing so, yields

$$\begin{aligned} \frac{\partial J}{\partial u_{ij}} = & m u_{ij}^{m-1} \sum_{s=1}^K v_{ij}^q d_{ijs} \\ & + \gamma \left(\sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \mathcal{S}} \sum_{l=1, l \neq i}^C \kappa_{jk} m u_{ij}^{m-1} u_{kl}^m + \sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \mathcal{N}} \rho_{jk} m u_{ij}^{m-1} u_{ki}^m \right) \\ & - 2\alpha \sum_{r=1}^N u_{ir} - \lambda_j = 0 \end{aligned} \quad (51)$$

Under the assumption that the membership values do not change significantly between consecutive iterations, (51) reduces to

$$u_{ij} = \left(\frac{\lambda_j + 2\alpha N_i}{m \left(\sum_{s=1}^K v_{ij}^q d_{ijs} + \gamma \left(\sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \mathcal{S}} \sum_{l=1, l \neq i}^C \kappa_{jk} u_{kl}^m + \sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \mathcal{N}} \rho_{jk} u_{ki}^m \right) \right)} \right)^{\frac{1}{m-1}} \quad (52)$$

where

$$N_i = \sum_{r=1}^N u_{ir} \quad (53)$$

is the fuzzy cardinality of cluster i .

Minimization of J with respect to the Lagrange multiplier produces

$$\frac{\partial J}{\partial \lambda_j} = \sum_{z=1}^C u_{zj} - 1 = 0 \quad (54)$$

Using (52) and (54), and solving for λ_j returns,

$$\lambda_j = \left(\frac{1 - \sum_{z=1}^C \left((2\alpha N_z) / H_{zi} \right)^{\frac{1}{m-1}}}{\sum_{z=1}^C (1/H_{zi})} \right)^{m-1}. \quad (55)$$

where

$$H_{ij} = m \left(\sum_{s=1}^K v_{ij}^q d_{ijs} + \gamma \left(\sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \mathcal{S}} \sum_{l=1, l \neq i}^C \kappa_{jk} u_{kl}^m + \sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \mathcal{N}} \rho_{jk} u_{ki}^m \right) \right).$$

Substituting λ_j from (55) in equation (52), it can be shown that the update equation for the membership of point \mathbf{x}_j in cluster i becomes:

$$u_{ij} = \left(H_{ij}^{\frac{1}{1-m}} \right) \left(\frac{1}{\sum_{z=1}^C H_{zi}^{\frac{1}{1-m}}} - \frac{\sum_{z=1}^C \left(\frac{H_{zi}}{2\alpha N_z} \right)^{\frac{1}{1-m}}}{\sum_{z=1}^C H_{zi}^{\frac{1}{1-m}}} + \left(\frac{1}{2\alpha N_k} \right)^{\frac{1}{1-m}} \right) \quad (56)$$

Optimization of (49) with respect to \mathbf{V} would yield an equation for updating the relevance weights. Since the rows of \mathbf{V} are independent, this optimization problem can be reduced to the following C simpler and independent problems:

$$\begin{aligned}
J_i(\phi_i, \mathbf{V}_i) &= \sum_{j=1}^N u_{ij}^m \sum_{s=1}^K v_{is}^q d_{ijs} \\
&+ \gamma \left(\sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \mathcal{S}} \sum_{l=1, l \neq i}^C \kappa_{jk} u_{ji}^m u_{kl}^m + \sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \mathcal{N}} \rho_{jk} u_{ji}^m u_{ki}^m \right) \\
&- \alpha \left[\sum_{j=1}^N u_{ij} \right]^2 - \phi_i \left(\sum_{s=1}^K v_{is} - 1 \right)
\end{aligned} \tag{57}$$

for $i = 1, \dots, C$,

In (57), \mathbf{V}_i is the i^{th} row of \mathbf{V} , and ϕ_i is a Lagrange multiplier used to incorporate the constraints in (48). By setting the gradient of J_i with respect to \mathbf{V}_i and ϕ_i to zero, we obtain

$$\frac{\partial J_i(\phi_i, \mathbf{V}_i)}{\partial \phi_i} = \left(\sum_{s=1}^K v_{is} - 1 \right) = 0 \tag{58}$$

$$\frac{\partial J_i(\lambda_i, \mathbf{V}_i)}{\partial v_{is}} = q v_{ik}^{(q-1)} \sum_{j=1}^N u_{ij}^m d_{ijs} - \phi_i = 0. \tag{59}$$

Solving (58) and (59) for the relevance weights v_{is} , we obtain

$$v_{is} = \frac{\left[1 / \sum_{j=1}^N u_{ij}^m d_{ijs} \right]^{1/(q-1)}}{\sum_{s=1}^K \left[1 / \sum_{j=1}^N (u_{ij})^m (d_{ij}^z)^2 \right]^{1/(q-1)}}. \tag{60}$$

Simplifying (60), v_{is} reduces to

$$v_{is} = \frac{1}{\sum_{z=1}^K \left(\tilde{D}_{is} / \tilde{D}_{iz} \right)^{1/(q-1)}}, \tag{61}$$

where $\tilde{D}_{is} = \sum_{j=1}^N u_{ij}^m d_{ijs}$ is the measure of dispersion for the i^{th} cluster along the s^{th} dimension, and $\sum_{k=1}^K \tilde{D}_{ik}$ is the cumulative dispersion of the i^{th} cluster. This relation

implies that the more compact the i^{th} cluster is along the s^{th} dimension (smaller \tilde{D}_{is}), the higher the relevance weight, v_{is} will be for the s^{th} feature.

In (61), the discrimination exponent $q \in (1, \infty)$ determines how much discrimination occurs between the relevance weights of different features subsets. For large values of q , there is little or no discrimination. For small values, there is greater discrimination.

Minimization of J with respect to the prototype parameters depends on the choice of d_{ijs} . Since each partial distance is treated independently from the others (i.e. disjoint feature subsets), the objective function in (49) can be decomposed into K independent problems:

$$\begin{aligned}
J_s = & \sum_{i=1}^C \sum_{j=1}^N u_{ij}^m \sum_{s=1}^K v_{is}^q d_{ijs} \\
& + \gamma \left(\sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \mathcal{N}} \sum_{i=1}^C \sum_{l=1, l \neq i}^C \kappa_{jk} u_{ji}^m u_{kl}^m + \sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \mathcal{N}} \sum_{i=1}^C \rho_{jk} u_{ji}^m u_{ki}^m \right) \\
& - \alpha \sum_{i=1}^C \left[\sum_{j=1}^N u_{ij} \right]^2, \text{ for } s = 1, \dots, K.
\end{aligned} \tag{62}$$

Each J_s would be optimized with respect to a different set of prototype parameters. For instance, if d_{ijs} is chosen as the Euclidean distance, the update equation for the centers, c_{is} , of subset s would be the same as the FCM [40]. That is, the center for feature subset, s , would be updated using

$$\mathbf{c}_{is} = \frac{\sum_{j=1}^N (u_{ij})^m \mathbf{x}_{js}}{\sum_{j=1}^N (u_{ij})^m}, \tag{63}$$

where \mathbf{x}_{js} includes only the s^{th} feature components of data sample \mathbf{x} . Similarly, if d_{ijs} , is the weighted Mahalanobis distance, minimization of J_s , would yield update equations for the centers and covariance matrices as in the GK algorithm [48]. That is, the centers would be updated using (63), and the covariance matrix for feature

subset s would be updated using

$$\mathbf{C}_{is} = \frac{\sum_{j=1}^N (u_{ij})^m (\mathbf{x}_{js} - \mathbf{c}_{is})(\mathbf{x}_{js} - \mathbf{c}_{is})^T}{\sum_{j=1}^N (u_{ij})^m}. \quad (64)$$

The constant γ in (49) is selected to allow balance between the sum of intra-cluster distances and the number of constraints. That is, in each iteration we update γ using

$$\gamma = \frac{N}{M} \frac{\sum_{k=1}^C \sum_{i=1}^N (u_{ik})^m d_{ijs}}{\sum_{k=1}^C \sum_{i=1}^N u_{ik}^m}, \quad (65)$$

where M is the number of pairwise constraints. Similarly, the agglomeration constant, α , is selected to allow balance between the sum of intra-cluster distances and the regularization term. That is, in each iteration k , we update α using

$$\alpha(k) = \eta_0 \exp(-k/\tau) \frac{\sum_{i=1}^C \sum_{j=1}^N u_{ij}^m \sum_{s=1}^K v_{is}^q d_{ijs}}{\sum_{i=1}^C [\sum_{j=1}^N u_{ij}]^2} \quad (66)$$

The CC algorithm is summarized below:

The Constrained Clustering Algorithm

Fix the maximum number of clusters $C = C_{max}$;
Fix m , $m \in (1, \infty)$;
Fix the discrimination exponent q , $q \in (1, \infty)$;
Initialize iteration counter $k = 0$;
Initialize the centers and the fuzzy C partition $\mathbf{U}^{(0)}$;
Initialize all the relevance weights to $1/K$;
Compute initial cardinalities N_i for $1 \leq i \leq C$ using (53);
Repeat
 Compute d_{ijs} for $1 \leq i \leq C$,
 $1 \leq j \leq N$, and $1 \leq s \leq K$;
 Update $\alpha(k)$ and γ using (66) and (37);
 Update the relevance weights matrix \mathbf{V} using (61);
 Compute D_{ij}^2 using (47);
 Update the partition matrix $\mathbf{U}^{(i)}$ using (56);
 Compute the cardinality N_i for $1 \leq i \leq C$ using (53);
 If $(N_i < \epsilon_1)$ discard cluster β_i ;
 Update the number of clusters C ;
 Update the prototype parameters;
 $k = k + 1$;
Until(centers and prototype parameters stabilize)

B The Adaptive Constrained Clustering (ACC) Algorithm

The ACC algorithm is an adaptive modification to the Constrained Clustering algorithm outlined in Section A. The adaptive properties of the algorithm are defined by both the search for the optimal number of clusters and by considering the constraint selection/satisfaction method. In particular, the ACC algorithm utilizes a method of competitive agglomeration to merge similar clusters and the partial supervision information to split clusters. In the CC algorithm, clustering begins by overestimating the number of clusters. Then the clusters begin competing for points and clusters are pruned as they become empty. Unfortunately, the process can be hindered by many factors such as, the structure of the data, the initialization of the clusters, and the value of the agglomeration constant. These factors can cause a sudden, nonrecoverable drop in the number of clusters and an optimal solution may not be possible.

In Figure 1, we provide an illustrative example of this potential drawback. A scatter plot of the 2-dimensional data used in the example is given in Figure 1(a). This data set consists of two clusters, circled in Figure 1(b). The initialization of the CA algorithm, Figure 1(c), overestimates the number of clusters and uses $c = 9$. In this Figure, each point's color indicates to which cluster it belongs. Figure 1(d) displays the results of the CA after 25 iterations, where the number of clusters has reduced to 6 and the agglomeration process has merged the small, circular cluster with a portion of the larger cluster. Figure 1(f) shows the final results of the CA clustering. Due to the merging of the two clusters in previous steps, the algorithm cannot recover from merging the incorrect clusters. This is because the CA cannot increase the number of clusters.

The proposed ACC algorithm attempts to alleviate this problem using the available partial supervision information. In particular, if during the iteration process,

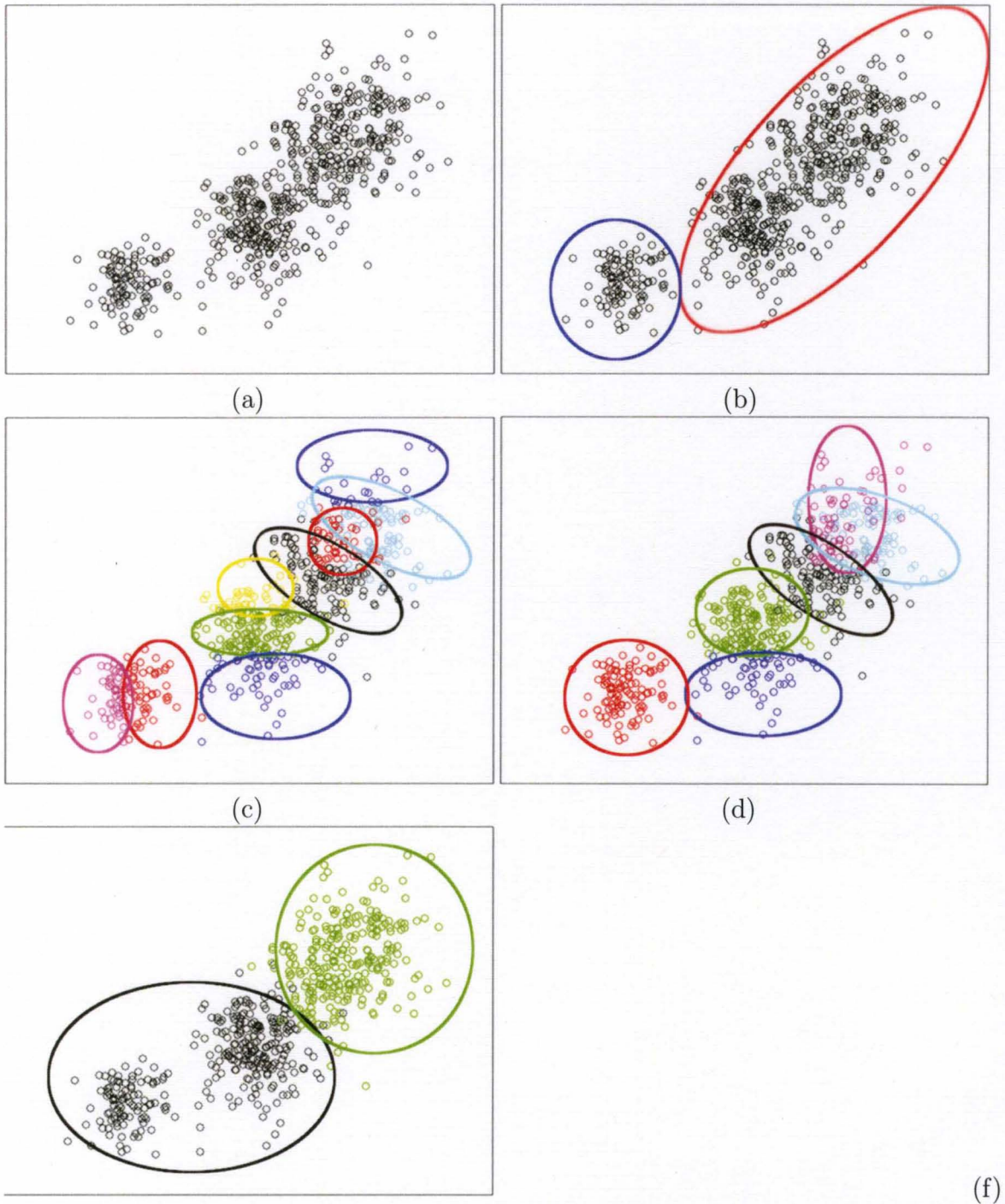


Figure 1. An illustrative example of over-agglomeration. (a) A 2-D data set with two clusters. (b) The ground truth of the two clusters. (c) Initialization of the CA algorithm with a over estimation of clusters $C_{max} = 9$. (d) After a few iterations, the number of clusters has reduced to 6, where the agglomeration merges two clusters inconsistently with the ground truth. (f) Upon convergence, portions of two distinct clusters have merged, yielding unfavorable clustering results. The CA algorithm cannot increase the number of clusters and recover from this local minima.

there exists a set of unsatisfied *should not*-link constraints, we increase the number of clusters to allow the satisfaction of the imposed constraints. This grants the ACC algorithm an adaptive feature where the number of clusters is allowed not only to contract but also to expand. More specifically, during clustering, the ACC algorithm inspects the current cluster distributions against the current set of *should not*-link constraints. When an unsatisfied constraint is discovered, the algorithm splits the pair of points creating a new cluster. The furthest point from the center of the old cluster is used to create a new cluster. The distances of all points assigned to the split clusters are recalculated using the two cluster centers and the two clusters are populated with points that are closest to their center. This method of cluster expansion not only assists in helping recover from over-agglomeration and finding the optimal number of clusters, but also in satisfying *should not*-link constraints.

In particular, the ACC algorithm uses *should-not* link constraints to split the clusters and guide the ACC algorithm to find the optimal number of clusters resulting in a dynamic algorithm where the number of clusters could shrink (using competitive agglomeration) or expand (using the splitting of clusters with unsatisfied *should-not* link constraints). The resulting algorithm, called Adaptive Constrained Clustering (ACC), is summarized below.

The Adaptive Constrained Clustering Algorithm

```

Fix the maximum number of clusters  $C = C_{max}$ ;
Fix  $m, m \in (1, \infty)$ ;
Fix the discrimination exponent  $q, q \in (1, \infty)$ ;
Initialize iteration counter  $k = 0$ ;
Initialize the centers and the fuzzy  $C$  partition  $\mathbf{U}^{(0)}$ ;
Initialize all the relevance weights to  $1/K$ ;
Compute initial cardinalities  $N_i$  for  $1 \leq i \leq C$  using (53);
Repeat
    Repeat
        Compute  $d_{ijs}$  for  $1 \leq i \leq C$ ,
             $1 \leq j \leq N$ , and  $1 \leq s \leq K$ ;
        Update  $\alpha(k)$  and  $\gamma$  using (66) and (37);
        Update the relevance weights matrix  $\mathbf{V}$  using (61);
        Compute  $D_{ij}^2$  using (47);
        Update the partition matrix  $\mathbf{U}^{(i)}$  using (56);
        Compute the cardinality  $N_i$  for  $1 \leq i \leq C$  using (53);
        If  $(N_i < \epsilon_1)$  discard cluster  $\beta_i$ ;
        Update the number of clusters  $C$ ;
        Update the prototype parameters;
         $k = k + 1$ ;
    Until(centers and prototype parameters stabilize)
    Repeat
        If a should-not link constraint is violated;
        Create two new centers from points violating constraint;
        Assign points surrounding the new centers to the newly
            formed clusters;
    Until(no constraints are violated)
Until(no new constraints are created)

```

The splitting process of the ACC algorithm is illustrated in Figure 2. In Figure 2(a), we display the results of the CA algorithm from the previous example (shown in Figure (1)). The red line links the two sample points that were selected for a *should-not* link constraint. Figure 2(b) displays the results from the split of the cluster containing the violated constraint. The results after resuming the CA with 3 clusters are displayed in Figure 2(c). As can be seen, the ACC algorithm has converged to the correct partition.

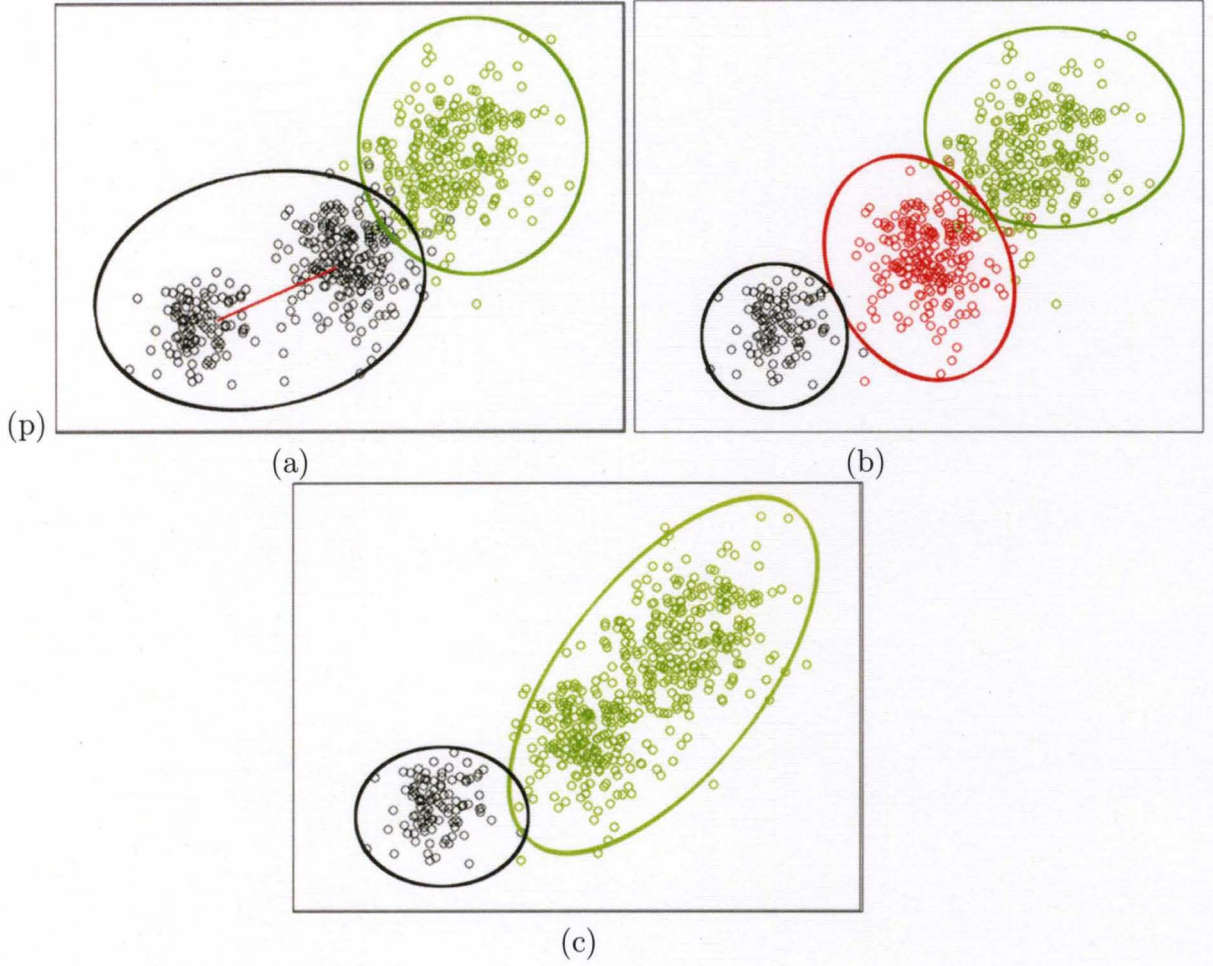


Figure 2. Illustration of the cluster splitting process in the ACC algorithm. (a) Partition resulting from the CA algorithm and selection of a *should-not* link constraint. (b) Splitting of the black cluster to satisfy the should not link constraint. (c) Final partition where the ACC has recovered from the over-agglomeration.

C Constraint Selection

The CC algorithm is a data partitioning method that can incorporate pairwise constraints to guide the optimization process. These constraints are derived from prior knowledge about the given data set and can be extracted from multiple sources. For instance, an interactive environment with users could use various methods of relevance feedback to create constraints between objects, Similarly, information could be retrieved from associated contextual information or metadata, or even from just a few labeled samples.

In the Chapter V, we show that for the database categorization application, contextual information from geographical, temporal, and semantic textual information can be used to adaptively create constraints.

D Computational Complexity

The ACC algorithm uses a single objective function designed to overcome some of the limitations observed in prototype-based clustering. One drawback to providing additional functionality is the additional computational complexity of the resulting algorithm. During the clustering process, each iteration of the ACC consists of several steps resulting in an updated partition of the given data set. The computational complexity of the ACC algorithm is on the order of $O(NCS + NCM + NC)$, where N is the number of data points, C is the total number of clusters, M is the total number of constraints and S is the number of feature subsets. The computation of the distances in (47), takes place in NCS time. Calculating the balancing constraints (66) and (65), and the agglomeration process are all performed in $O(NC)$ time. Updating the memberships is on the order of $O(NCM)$, which takes into account the M constraints utilized by the system. Therefore, the total computational complexity of the ACC over K iterations can be estimated as $O(KNC(S + M + 1))$.

E Convergence Properties

The ACC algorithm minimizes the objective function defined in (49) subject to the constraints in (46) by means of alternating optimization. In other words, in each iteration the algorithm first learns a set of memberships U , under the assumption that the cluster centers V are fixed. Then the algorithm updates each cluster center under the assumption that the memberships are fixed [93]. Methods which utilizes the alternating optimization are known as gradient descent algorithms [94].

These algorithms are first-order optimization algorithms designed to seek the local minimum of a given function. Therefore, algorithms which seek to minimize an objective function using alternating optimization are guaranteed to converge to at least a local minima, or saddle point [95, 96]. These algorithms are set to terminate when the alternating parameters stabilize. That is, the process starts with a initial guess for V or U and continues until successive iterations of the parameters differ by a minimal amount (i.e. $\|U_{k+1} - U_k\| < \epsilon$ where k is the iteration number). Typically, the rate of convergence is unknown and in practice we terminate the algorithm after 100 iterations to limit execution time.

CHAPTER IV

EXPERIMENTAL EVALUATIONS

A Introduction

In this chapter we illustrate the performance of the ACC using several synthetic data sets. The ACC algorithm is compared with the SCADCA algorithm [77], and the PCCA algorithm [26]. These algorithms are outlined in Chapter II, sections C and D.2 respectively. The remainder of this section provides an overview of the data sets used in these experiments and defines the performance measures used for evaluation. Then, Section C, compares the computational complexity of the algorithms and addresses the issue of initialization with regards to the resulting performance. In Section B, we use the various performance measures to compare the partitioning performance of the algorithms, taking into account the addition of constraints.

1 Data Sets

The different algorithms are compared using a collection of synthesized data sets (see Appendix A). Using synthetic data allows for an easier method of tracking the changes to the data during the clustering process. Several data sets were generated in order to test the clustering process and analyze the results of the ACC. These data sets are summarized in Table 1 and shown in Figure 3. Each point's features are represented simply as their Cartesian points in the two dimensional space. For the purpose of constraint selection, the data is labeled so that the class for each data point is known. These labels are only used during the constraint selection process.

TABLE 1

Data sets used in the algorithm comparison.

Data set	No. of Points	No. of Clusters
1	200	2
2	200	3

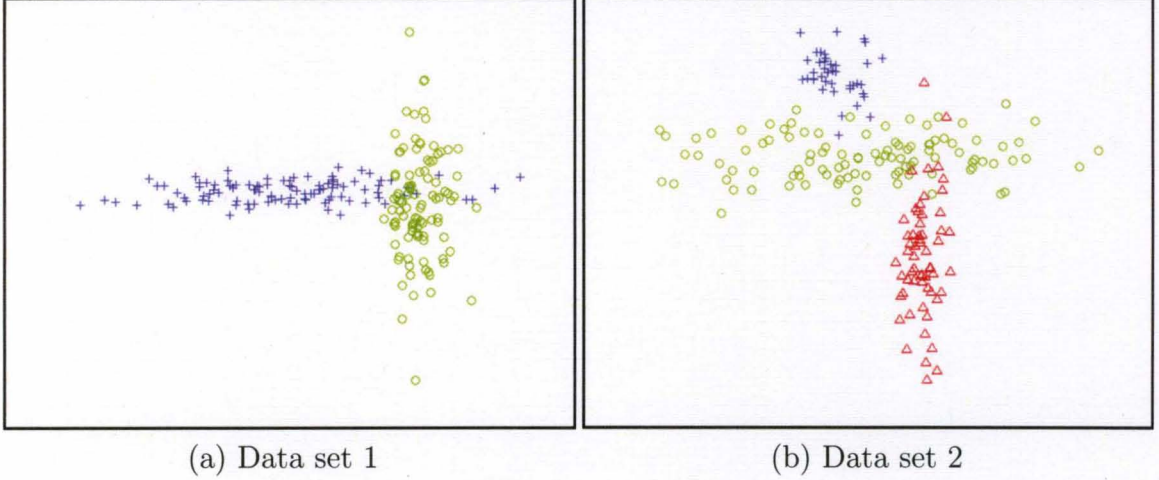


Figure 3. The synthetic data sets used to evaluate and compare the algorithm. Points from each cluster are represented by a different color and symbol.

2 Performance Measures

The ACC algorithm and the other algorithms used for comparison, generate a $C \times N$ fuzzy partition matrix $\mathbf{U} = [u_{ij}]$, $i = 1, \dots, C$; $j = 1, \dots, N$, where $u_{ij} \in [0, 1]$ is the fuzzy membership degree of the j^{th} data point in the i^{th} cluster. For the purposes of testing and constraint creation, we assume that the generated data is labeled. That is, the ground truth for each data set is known and can be represented by a partition matrix $\mathbf{U}^{(T)} = [u_{ij}^{(T)}]$, where $u_{ij} \in \{0, 1\}$ are crisp memberships.

In [97], many measures that compare two partitions are given. Three of these measures will be used to compare the clustering efficiency of the ACC algorithm against the PCCA algorithm, and the SCADCA algorithm [77]. The CA algorithm [42] was excluded from these comparisons because the PCCA algorithm reduces to

the CA when no constraints are used.

The comparison of two partition matrices, $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$, begins by computing their coincidence matrices $\Psi^{(1)}$ and $\Psi^{(2)}$ where

$$\Psi = [\psi_{jk}], 1 \leq j, k \leq N, \quad \text{and} \quad \psi_{jk} = \sum_{i=1}^C u_{ij} u_{ik}.$$

Then, a 2×2 contingency table is computed as in Table 2 where

$$\begin{aligned} N_{SS}(\Psi^{(1)}, \Psi^{(2)}) &= \sum_{j=2}^N \sum_{k=1}^{j-1} \psi_{jk}^{(1)} \psi_{jk}^{(2)}, \\ N_{SD}(\Psi^{(1)}, \Psi^{(2)}) &= \sum_{j=2}^N \sum_{k=1}^{j-1} \psi_{jk}^{(1)} (1 - \psi_{jk}^{(2)}), \\ N_{DS}(\Psi^{(1)}, \Psi^{(2)}) &= \sum_{j=2}^N \sum_{k=1}^{j-1} (1 - \psi_{jk}^{(1)}) \psi_{jk}^{(2)}, \\ N_{DD}(\Psi^{(1)}, \Psi^{(2)}) &= \sum_{j=2}^N \sum_{k=1}^{j-1} (1 - \psi_{jk}^{(1)}) (1 - \psi_{jk}^{(2)}). \end{aligned}$$

In the above, the indices S and D stand for "same cluster" and "different clusters" respectively.

TABLE 2

Contingency table.

	$\psi_{jk}^{(2)} = 1$	$\psi_{jk}^{(2)} = 0$	Σ
$\psi_{jk}^{(1)} = 1$	N_{SS}	N_{SD}	$N_{S.}$
$\psi_{jk}^{(1)} = 0$	N_{DS}	N_{DD}	$N_{D.}$
Σ	$N_{.S}$	$N_{.D}$	$N_{..}$

Using the contingency table, the *Rand statistic* (Rand), the *Jaccard coefficient* (Jacc), and the *Folkes-Mallows index* (FMI) to compare each generated partition $\mathbf{U}^{(cl)}$ to the ground truth partition $\mathbf{U}^{(T)}$. These indices are defined as:

$$Q_{Rand}(\Psi^{(cl)}, \Psi^{(T)}) = \frac{N_{SS} + N_{DD}}{N_{..}},$$

$$Q_{Jacc}(\Psi^{(cl)}, \Psi^{(T)}) = \frac{N_{SS}}{N_{SS} + N_{SD} + N_{DS}},$$

$$Q_{FMI}(\Psi^{(d)}, \Psi^{(T)}) = \frac{N_{SS}}{\sqrt{(N_{SS} + N_{SD})(N_{SS} + N_{DS})}},$$

All of the above measures provide larger values when the two partitions are more similar.

B Experimental Results

For the remainder of this dissertation, we assume that all necessary and relevant features used during computations have been precomputed and are available in their entirety. All experiments were then computed using a 3.0GHz Pentium Xeon processor with 3.0GB of memory. In the following experiments, the maximum number of clusters was set to $C_{max} = 20$, the fuzzifier was set to $m = 1.5$, and the discriminant exponent was set to $q = 2.0$. The supervision constant, γ , is calculated using (65). For the agglomeration constant, α in (66), $\eta_0 = 2.0$ and $\tau = 20$. For the purposes of constraint creation, the ACC algorithm compares the current cluster distribution, every 25 iterations or one epoch, to the labels provided with the data. During the comparison, if a candidate *should* or *should-not* link constraint is found it is then added to the current list of constraints. For the following experiments, up to 5 new constraints were incorporated at each interval during the clustering process with precedence given towards the creation of *should-not* link constraints.

The synthetic data sets are displayed in Figure 3. The colors of each symbol indicate to which cluster they belong. Each point is represented as a symbol, these symbols represent different contextual descriptors for each point. That is, clusters can be formed using two methods. First by a density based notion to clusters, and second by defining the cluster using contextual descriptors.

In Figure 4, we illustrate the initial partitions of Data Set 1. Figure 4(a) shows the ground truth of the data set, with all the points labeled with a '+' belonging to the blue cluster and the points labeled with a 'o' belonging to the green. The results of

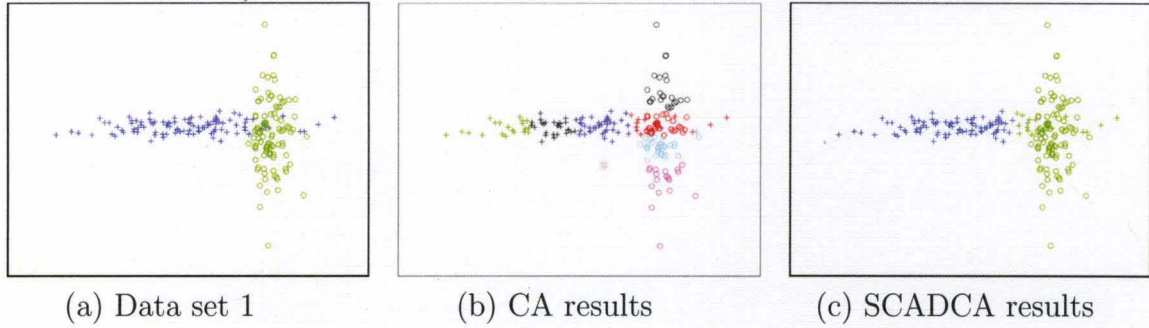


Figure 4. Initialization of Data set 1. The shape of each symbol refers to the contextual meaning of each point, and the color represents the assigned cluster.

the CA algorithm which is used to initialize the PCCA algorithm are given in Figure 4(b), while Figure 4(c) presents the results of the SCADCA algorithm which is used to initialize the ACC algorithm. The main difference between the initializations is evident by observing the formation of the clusters in each data set. In Figure 4(b), there exist numerous, small, circular clusters which is primarily due to the PCCA algorithm's inability to take into account the structure of the data set. On the other hand, the SCADCA algorithm is able to determine the optimal number of clusters, using the learned feature relevance weights of each cluster (see Table 3).

TABLE 3

Feature relevance weights learned during initialization of the ACC algorithm.

Cluster	Horizontal Weight	Vertical Weight
1 (blue)	0.7239	0.2761
2 (green)	0.4183	0.5817

Figures 5(a-c) and 6(a-c) illustrate the effects of iteratively incorporating constraints into the PCCA and ACC algorithms respectively. After five epochs, the PCCA algorithm is able to reduce the number of clusters from six to four. Unfortunately, at this point the agglomeration stalls in the PCCA through the remaining epochs (Figure 5(b-c)) because the formulated clusters have become well defined with

respect to the defined distance measure.

The intermediate results of the ACC algorithm are displayed in Figure 6. Here, the effects of the cluster splitting paradigm is illustrated in Figure 6(a), where the vertical cluster (green) has been split into two clusters based on the given constraint set. In this case, the splitting allows the ACC algorithm to avoid becoming trapped in a local minima. In Figure 6, the ACC algorithm is able to again reduce the partition to the optimal number of clusters, but in doing so misclassifies a number of points which leads to the creation of additional constraints. Finally, Figure 6(c) shows the resulting partition after 15 epochs. At this point the ACC algorithm has created three clusters in order to remove the misclassified points, '+', from the vertical (blue) cluster.

It is important to note that as the constraints are gradually added to both the PCCA and the ACC algorithms, the performance may decrease between epochs. In the ACC algorithm, this is primarily due to instances in which a cluster may be split and not quickly agglomerated between epochs. The comparison of accuracy results during the clustering process is illustrated in Figures 7(a-c). Note that using the Q_{rand} measure, the PCCA algorithm produces a more accurate partition initially, but the performance values over all epochs supports the assumption that the agglomeration process had stalled. As the ACC algorithm converged, the performance, based on the

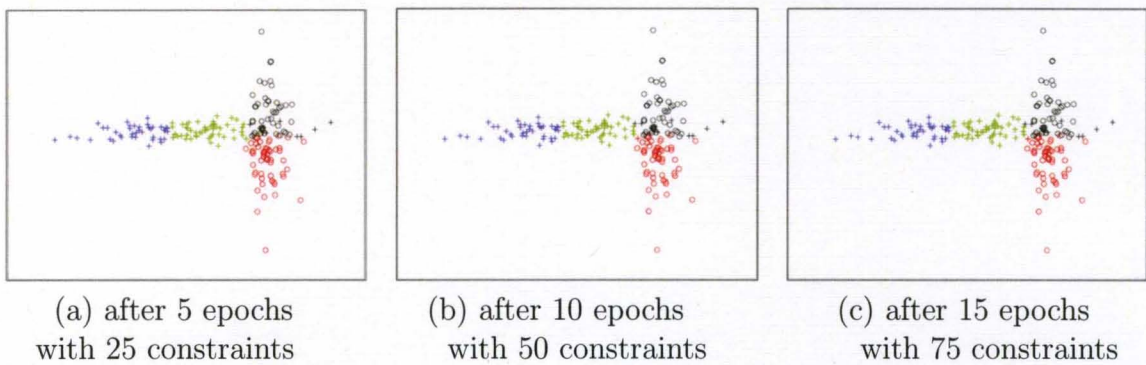


Figure 5. Intermediate results of the PCCA algorithm on Data Set 1

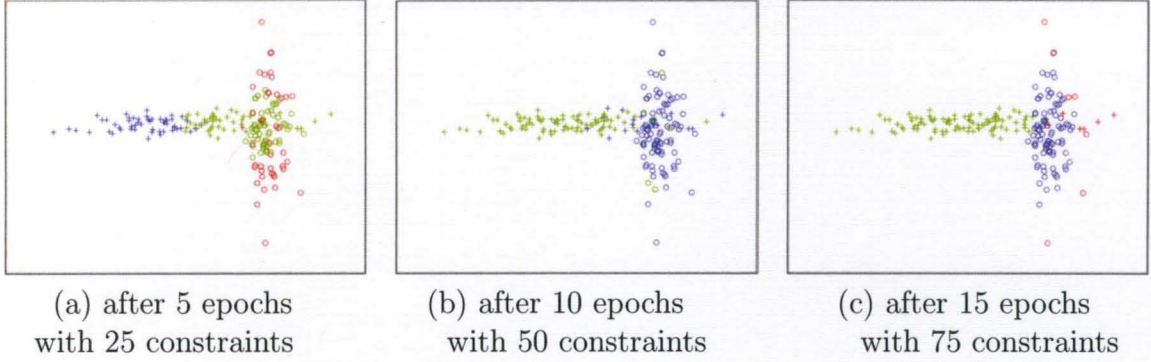


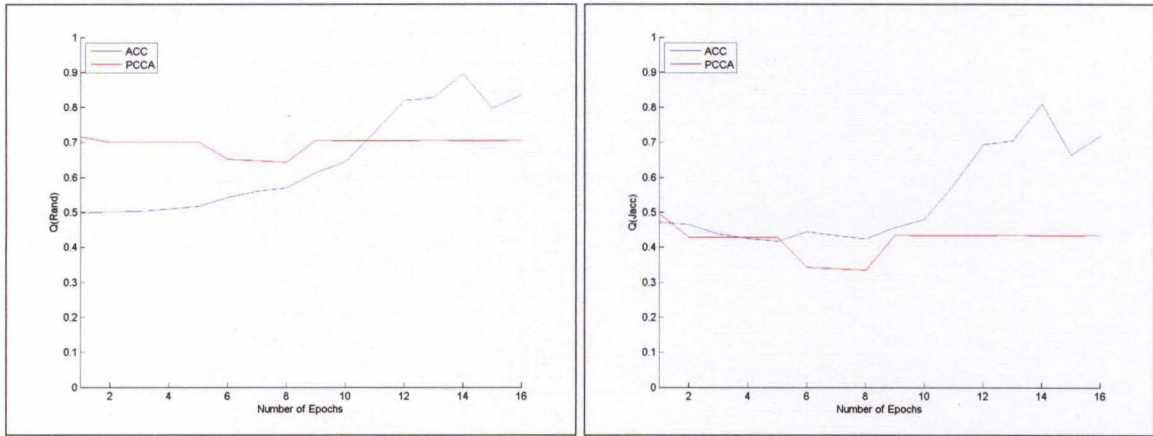
Figure 6. Intermediate results of the ACC algorithm on Data Set 1

ground truth, rose from the initial clustering and in all cases the ACC algorithm was able to overtake the PCCA algorithm and provide a more accurate partition.

Data set 2 illustrates the ability of the ACC algorithm to search for elliptical and spherical clusters simultaneously. After initialization (Figure 8(b)), the PCCA algorithm was able to only shift points between clusters, successfully merging one cluster, only to shift points to another cluster to maintain the total of five (Figures 9(a-c)). The ACC algorithm was initialized to three clusters (Figure 8(c)), and during the first epoch interval, agglomerated to two. The ACC algorithm then continued in Figures 10(a-c), shifting points to a new cluster over the last two epochs shown. These findings are evident from the performance evaluations (Figure 11). In the early epochs, the agglomeration's dominance was gradually decreased by adding the constraints.

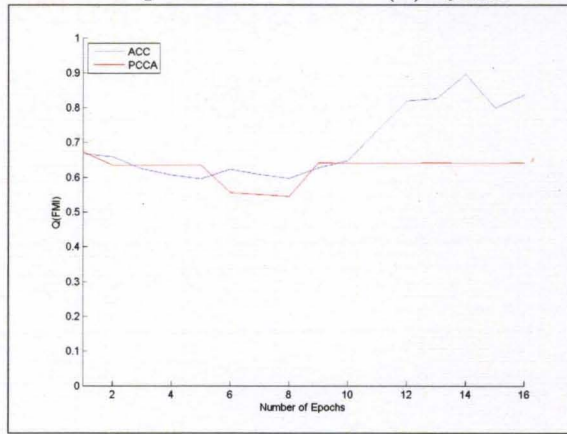
C Convergence Properties and Computational Complexity

Initialization can play an important role in the performance of an iterative clustering algorithm. The process of initialization varies with respect to the cluster prototypes, distance measures, and the application of the algorithm. In Figures 12(a-c), we demonstrate the effects of initialization on the the ACC, SCADCA, and PCCA algorithms. In these experiments, each algorithm was run 100 times on the points



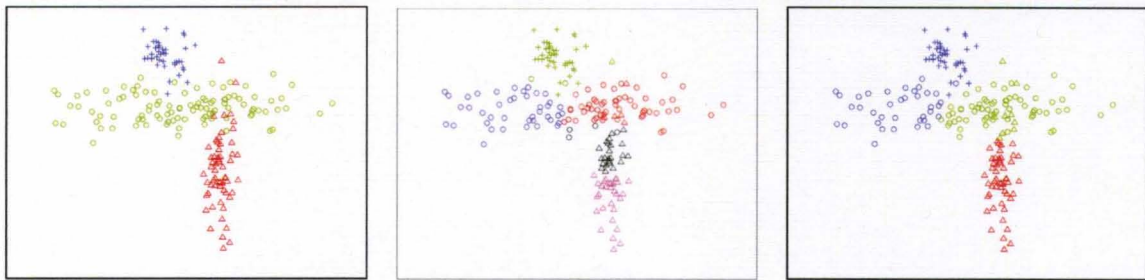
(a) Q_{Rand} vs. Number of epochs

(b) Q_{Jacc} vs. Number of epochs



(c) Q_{FMI} vs. Number of epochs

Figure 7. Performance evaluations for Data Set 1



(a) Data set 2

(b) CA results

(c) SCADCA results

Figure 8. Initialization of Data Set 2. The shape of each symbol refers to the contextual meaning of each point, and the color represents the assigned cluster.

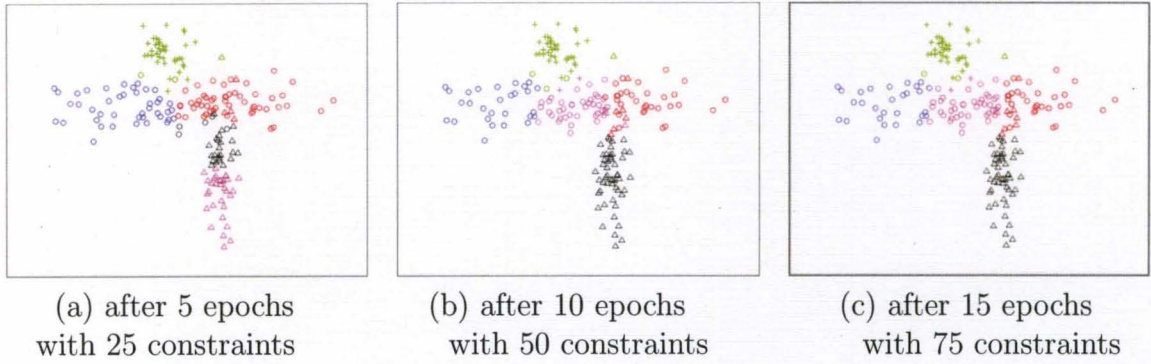


Figure 9. Intermediate results of the PCCA Algorithm on Data Set 2

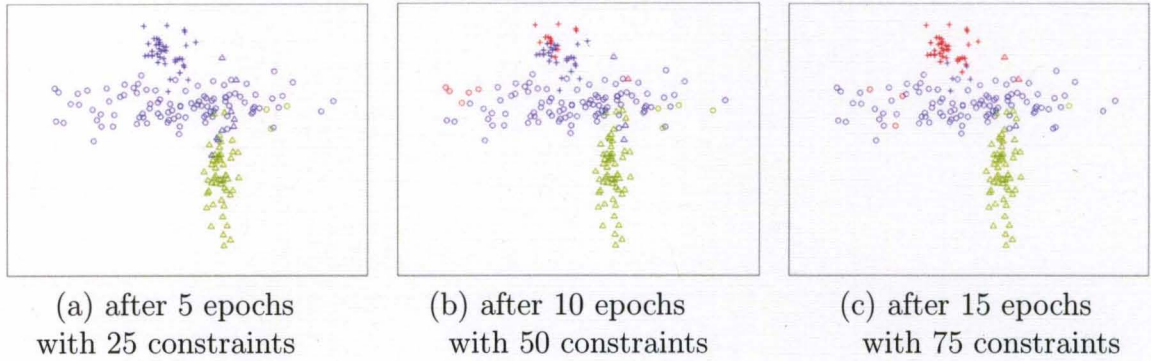


Figure 10. Intermediate results of the ACC algorithm on Data Set 2

in Data set 1 using different random points each run to initialize the clusters. In terms of constraint creation, the number of constraints created at each epoch was increased to 50. The results of the SCADCA algorithm are illustrated in Figure 12(a), which presents the average value and the standard variation of the objective function calculated over all passes of the algorithm. It is observed that on average, the SCADCA algorithm needed 55 iterations before convergence. In Figure 12(b), the PCCA algorithm suffers from a large amount of variance early in the clustering process which is attributed to the early agglomeration of clusters. The average number of iterations before convergence in the PCCA was calculated as 65 iterations. Finally, the ACC algorithm is illustrated in Figure 12(c). The ACC algorithm sees significant variations on the objective function in early iterations this again can be attributed to the agglomeration of clusters. Unlike the PCCA algorithm, on average the ACC

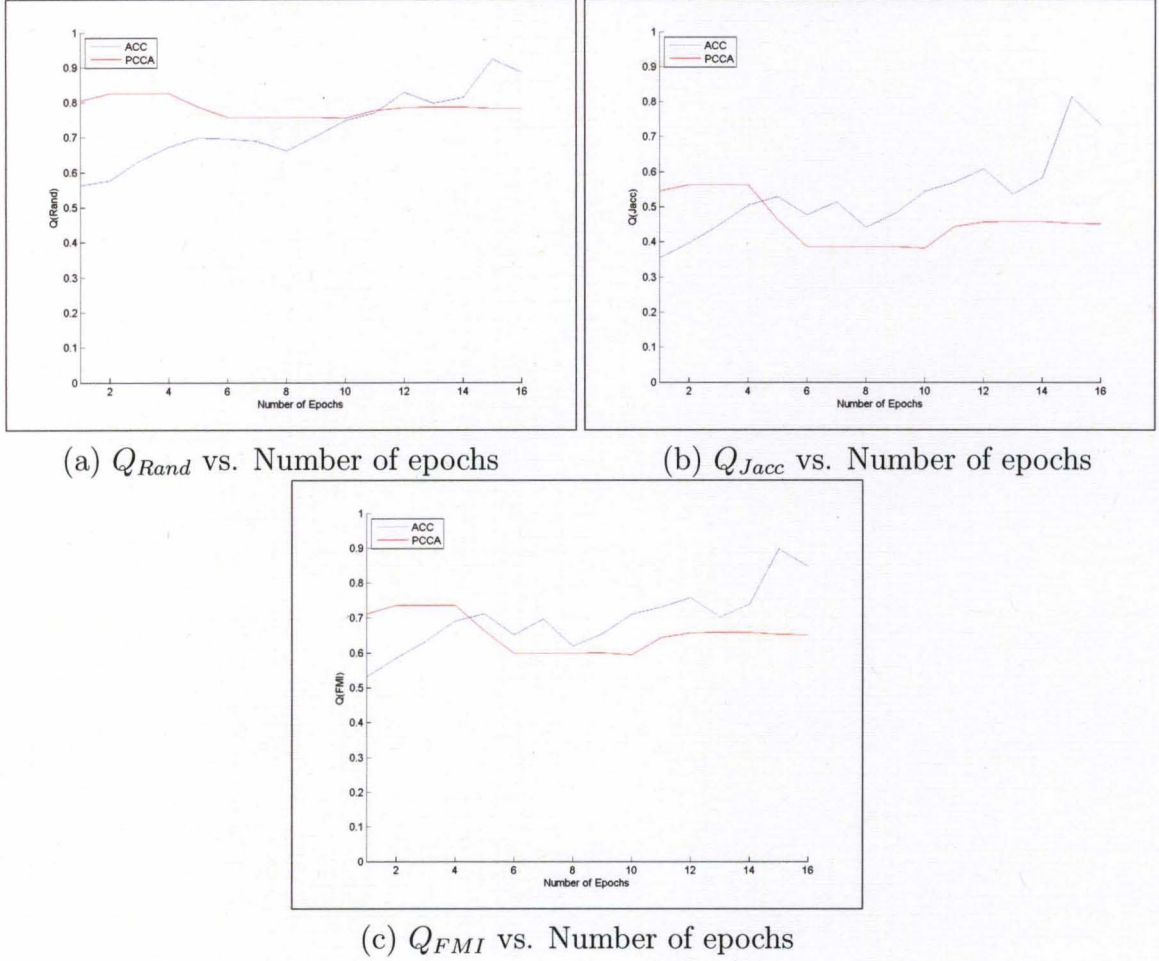


Figure 11. Performance evaluations for Data Set 2

algorithm sees a sharp decline early in the clustering process, due to the increase in the number of clusters using the splitting paradigm (outlined in Chapter III.§B). Figure 13 shows the number of clusters observed over 10 experiments with the ACC algorithm. These results illustrate the ACC algorithm's insensitivity to initialization, and despite different initializations with varying number of clusters, our approach is able to converge to the optimal number of clusters.

The computational complexity of the ACC algorithm can be estimated as $O(NCS + NCM + NC)$, where N is the number of data points, C is the total number of clusters, M is the total number of constraints and S is the number of feature subsets (refer to Chapter III.§D). Using Data set 1, the average observed

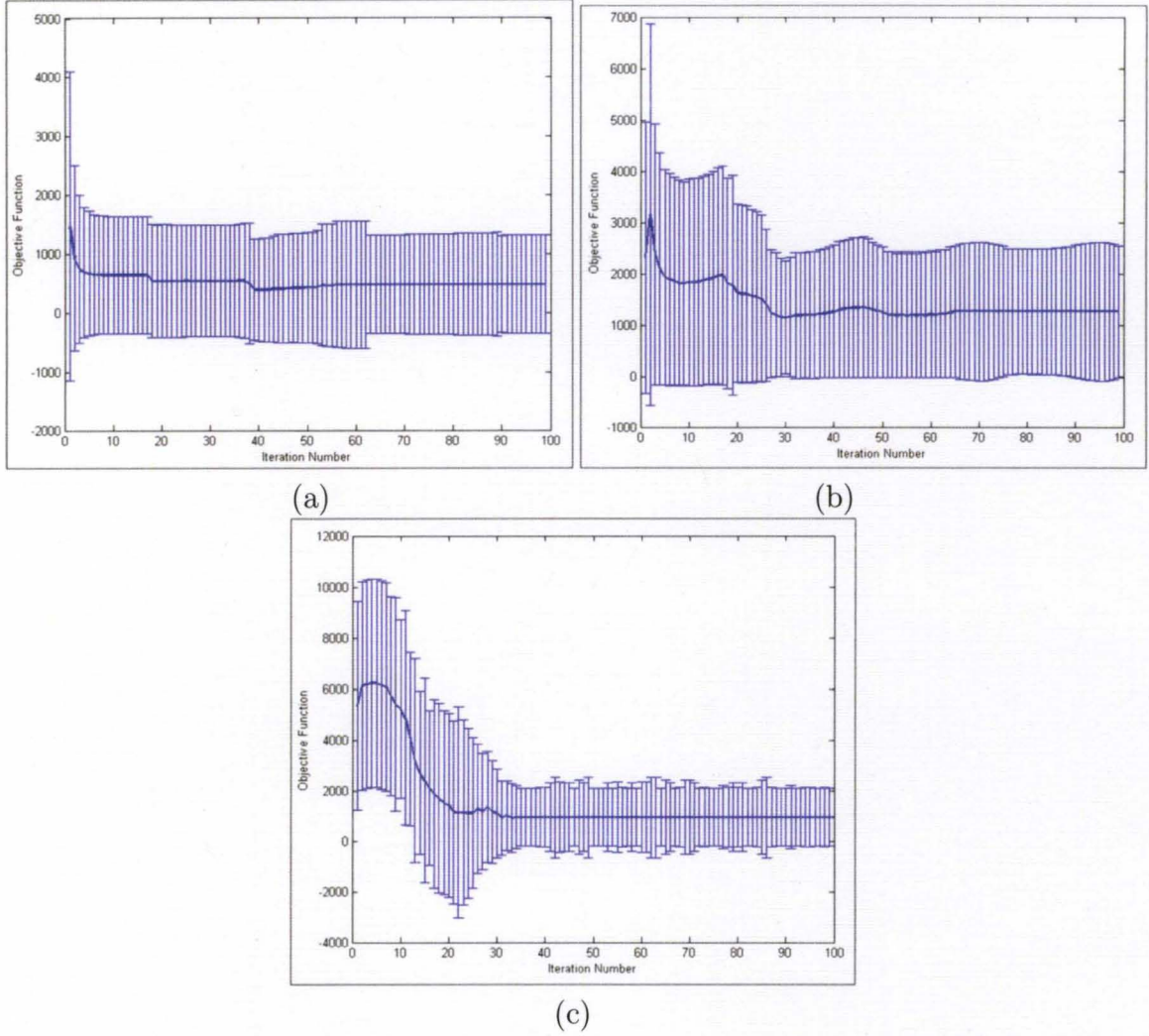


Figure 12. Evaluation of the objective function over 100 synthetic trials. (a) Results for the SCADCA algorithm (b) Results for the PCCA algorithm (c) Results for the ACC algorithm

running time for each algorithm is given in Table 4 and Table 5 provides the average running time for Data set 2. All three algorithms share some common traits, they all seek the optimal number of clusters using agglomeration. Where the SCADCA and the ACC share the ability to use feature subsets and assign relevance weights to each subset, while the PCCA and ACC algorithms both use partial supervision during the clustering process. Therefore, we would assume that the computational times of the algorithms would be on the order of the SCADCA, then the PCCA, and the ACC. In

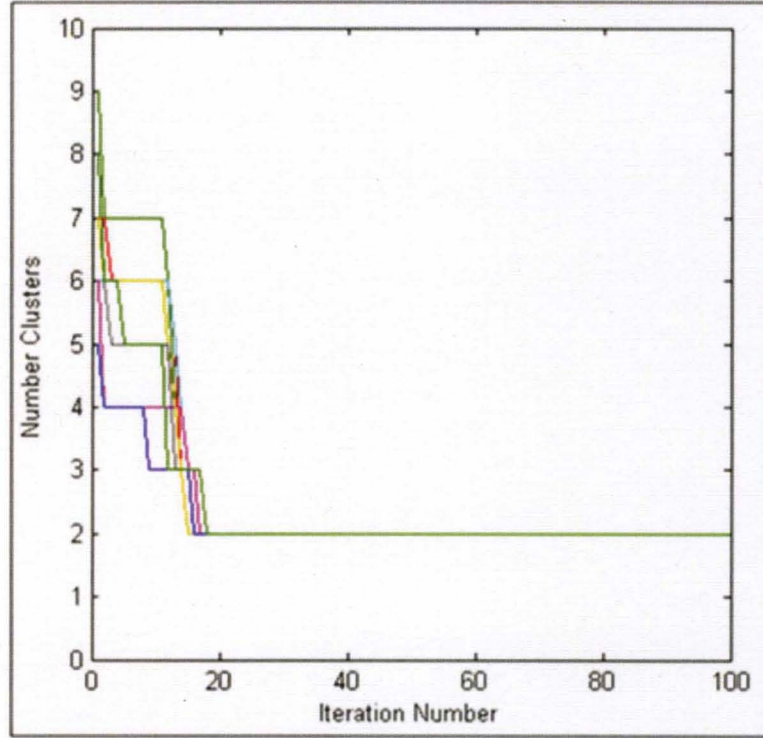


Figure 13. The number of clusters used per iteration in the ACC algorithm.

observation, the running time of the ACC algorithm is lower than that of the PCCA. The main factor that contributes to the ACC algorithm's improved computational performance over the PCCA can be attributed to fewer number of iterations

TABLE 4

The observed running time for each algorithm on sample Data set 1.

Algorithm	Run Time
ACC	5.06s
SCADCA	1.21s
PCCA	9.17s

In Section B, we demonstrated the ACC's ability to provide a more optimal partition from a single initialization. In Figure 14, we show the ACC algorithm's robustness towards initialization using the average performance measures described in Section A.2. As it can be seen, the ACC algorithm demonstrates superior levels of

TABLE 5

The observed running time for each algorithm on sample Data set 2.

Algorithm	Run Time
ACC	8.55s
SCADCA	1.14s
PCCA	11.05s

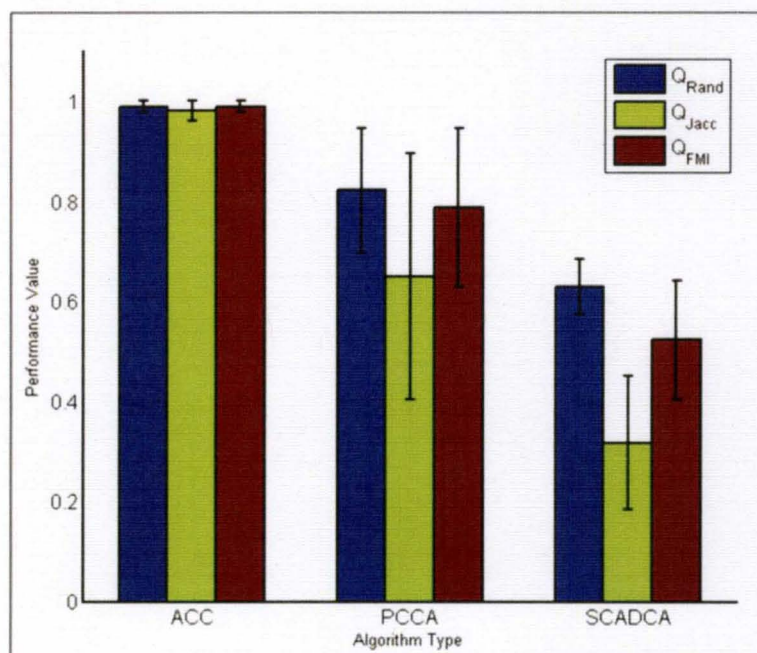


Figure 14. Average and standard deviation using the Q_{Rand} , Q_{Jacc} , and Q_{FMI} performance measures over 100 runs with different initialization. The low variance of the ACC indicates less sensitivity to different initializations.

partition validity for all performance measures, while maintaining a low variance. This means that the ACC algorithm converges consistently to the same optimal partition. On the other hand, The PCCA and SCADCA methods show varying levels of validity, with an accompanying increased level of variation in the results. This indicates more sensitivity to initialization.

CHAPTER V

DYNAMIC IMAGE DATABASE CATEGORIZATION AND VISUALIZATION USING ADAPTIVE CONSTRAINED CLUSTERING

A Motivations

The ability to capture, store, and view images has quickly become an everyday task in recent years. The quantity of images available to a user, either by personal image capture, or through secondary acquisition (i.e. Web communities) increases day by day. These communities, such as Flickr [2] or Panoramio [3], help demonstrate the scale of digital imagery available, and points toward the social and practical impact that viewing and interacting with images provides.

Image management researchers have taken many different paths to attempt to find meaningful, efficient methods to process images collections. Grouping, visualizing, and navigating image collections is a challenging task when the sizes range in the few thousands, and the scope of the task is immense when the collections number in the tens- or hundreds of thousands. Early approaches, utilized preexisting semantic keywords associated with each image and were found to have many difficulties, including tedious manual annotations and semantic ambiguity. These issues led to research in the area of content-based image retrieval, where visual descriptors were extracted to provide the means for image comparison. An issue known as the semantic gap [19] has caused substantial limitations to CBIR research. Recently, research directed

towards the combination of content and context retrieval has emerged [30, 31]. The hopes of these combinations is that the associated context will help to bridge the semantic gap and allow for more meaningful content retrieval.

The remainder of this chapter is organized as follows. First, content-based database categorization is described in Section B, and includes the feature extraction methods and clustering for categorization purposes. Next, Section C presents various methods of constraint creation using different types of contextual information. Finally, the experimental results are given in Section F, which consists of algorithm evaluations based on a sample data set using images that contain textual annotations.

B Content-Based Image Database Categorization

1 Feature Extraction

Regardless of the type of contextual information used for creating constraints, the content of the images must also be represented in a concise and efficient fashion. For the remainder of this thesis, each image is characterized by low-level visual feature subsets representing its content. The features used are MPEG-7 features [32] which are some of the most commonly used features in content-based image retrieval [18, 98].

Color Structure Descriptor (CSD): This descriptor represents an image using both its color distribution, based on color histograms, and the local spatial structure of the color by using a small structuring window. It maintains a count per instance of a particular color if found within the structuring element, as the element scans the image [32].

The CSD extraction is a three step process:

1. A 256-bin color histogram is accumulated (i.e. extracted) from an image that is mapped to the HMMD color space.

2. If the number of colors is less than 256, $N < 256$, bins are then unified to obtain a N -bin histogram.
3. The values of each bin are nonlinearly quantized in accordance with the statistics of color occurrence in typical consumer imagery.

Scalable Color Descriptor (SCD): The SCD is derived from a uniformly quantized, 256 bin color histogram taken from the HSV color space. The compiled histogram is then encoded using a Haar transform-based encoding scheme. The Haar transform is applied to four-bit integer values across the bins.

Natural image histograms tend to exhibit high levels of redundancy in adjacent bins, explained by the slight variation of colors caused by illumination and shadowing effects. Therefore, summing adjacent bins in pairs equates to producing a histogram with half the number of bins as the original. The binary representation of the Haar transform is scalable in terms of bin numbers and bit representation accuracy over a broad range of data rates. Typically for 256 bins, the highest 32 frequency components are needed for the experiments.

Homogenous Texture Descriptor (HTD): This feature uses Gabor descriptors proposed by Manjunath et al. [10] to represent the texture. Each image is filtered by 30 Gabor filters at 5 different scales and 6 orientations. The texture feature is represented by the average and standard deviation of each filtered image. It is believed that this representation can model the early visual processing of the human visual cortex [32].

Edge Histogram Descriptor (EHD): The design of this descriptor al-

lows for the representation of the spatial distribution, frequency, and directionality of the edges within each image. A simple edge detector is first used to identify edges and group them into five categories: vertical, horizontal, 45° diagonal, 135° diagonal, and the isotropic or non-edge. The local, global, and semi-local edge histograms are generated and concatenated to form a 150-dimensional feature vector [32].

2 Image Database Categorization Using Machine Learning Techniques

The goal of content-based image database categorization is to apply statistical learning methods to their low-level features, grouping the images into semantically meaningful categories. These categorization techniques could be used to summarize the data to provide adequate means to navigate the data set by providing an overview of the image collection. Database categorization could be achieved by using a supervised or unsupervised learning method. In the case where users are willing to provide labels for all images in a collection, supervised learning would be the method of choice. Practically, this approach is only viable for well-specified image collections. More specifically, a limited ontology may not be sufficient to categorize a large collection of generic photos. Therefore, in the case of large image collections, unsupervised learning or clustering, which does not require labeled data, requires little or no user intervention, and can group the collection into an optimum number of concepts or clusters, is the method of choice.

Figure 15(a-d) provides an example of how clustering can be beneficial in organizing and categorizing image collections. In Figure 15(a), a collection of images is summarized by 9 groups. Each image in Figure 15(a) represents a cluster of similar images. That is, each image is a semantic representative of a cluster that may contain

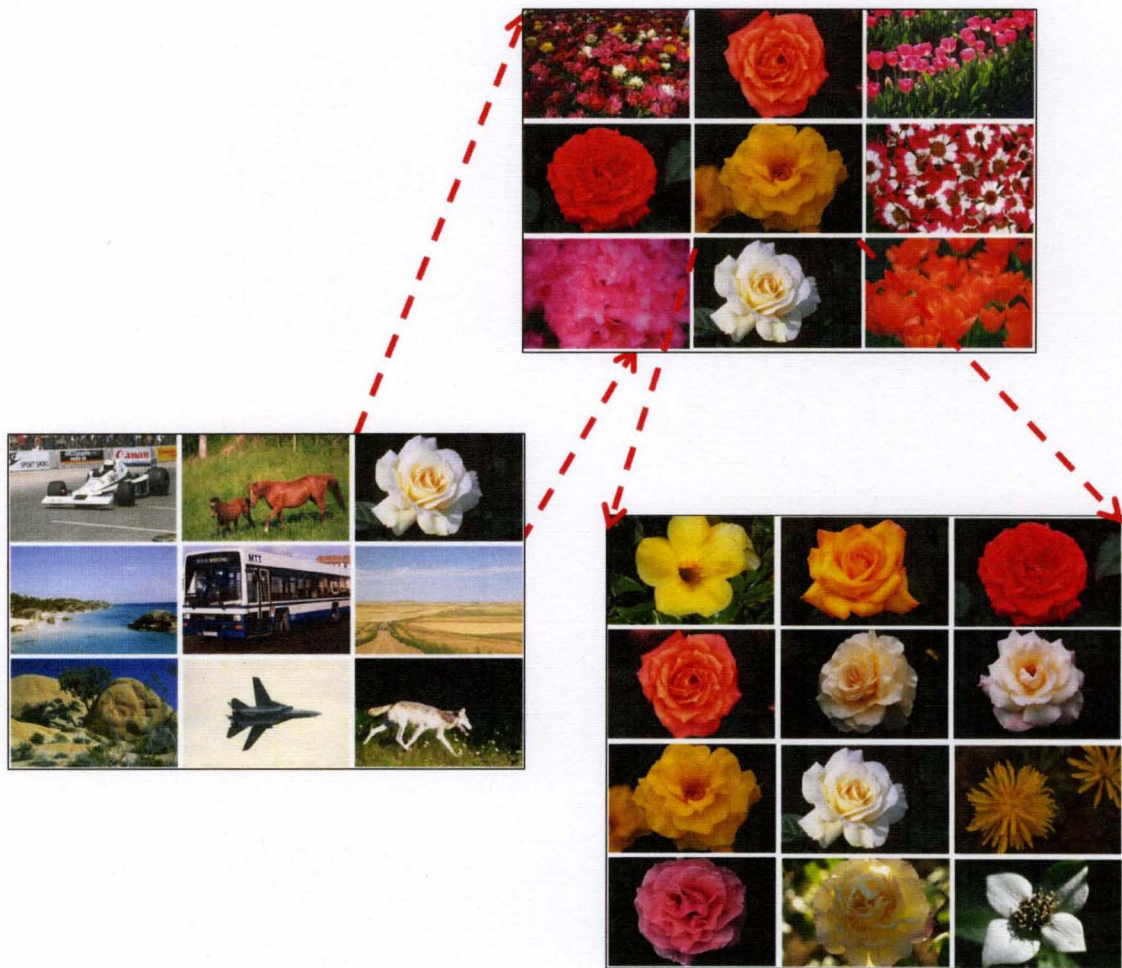


Figure 15. Illustrative example of cluster zooming. (a) An overview of the images in the data set. (b) The images contained in the cluster represented by the flower. (c) Images contained in a subsequent zoom level, continuing to navigate based on the selection of flowers.

many images. The 9 representative images provide a good overview of content. The user may then select a cluster of interest (e.g. flower) and an expanded view of the selected cluster becomes viewable. Dependent on the size of the cluster or clusters, this zooming process can be comprised of many different levels. If the resulting subset of images is still too large to view in its entirety, the process can re-cluster the current data to provide the user with a summarization of finer resolution, until the application is able to accommodate the current subset in its entirety.

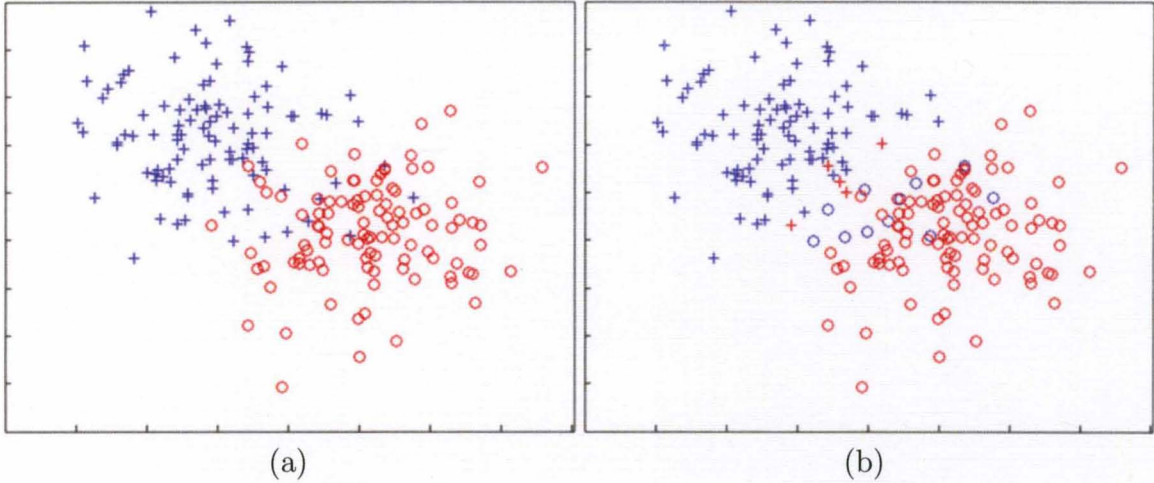


Figure 16. Sample data set demonstrating the need for partial supervision. (a) Known truth values for given data set. Marker color indicates cluster assignment, marker shape indicates class label. (b) Results of a typical unsupervised clustering algorithm. Note the misclassification of points in the overlapping region between clusters.

3 Image Database Categorization using the ACC algorithm

In Chapter III, we outlined the ACC algorithm which offers functionality and methods to overcome the main disadvantages of existing clustering algorithms. In Figure 16, we provide an illustrative example where obtaining meaningful results requires the use of partial supervision. Figure 16(a) shows the ground truth for the sample data set, in this example we know all points labeled with a '+' and 'o' belong to two separate clusters, while the color of each point indicates their cluster assignment (i.e. all blue points belong to cluster 1, and all red points belong to cluster 2). The results of a typical unsupervised clustering algorithm are given in Figure 16(b). In this example, the algorithm is able to create a partition with two clusters which is optimal. However, the unsupervised algorithm is unable to correctly classify some of the points in the overlapping area. In this case, it is known that these points are misclassified, which is inferred from the ground truth of the data set.

The ACC algorithm accounts for the necessity of limited partial supervision by incorporating pairwise constraints into the clustering process. These constraints are

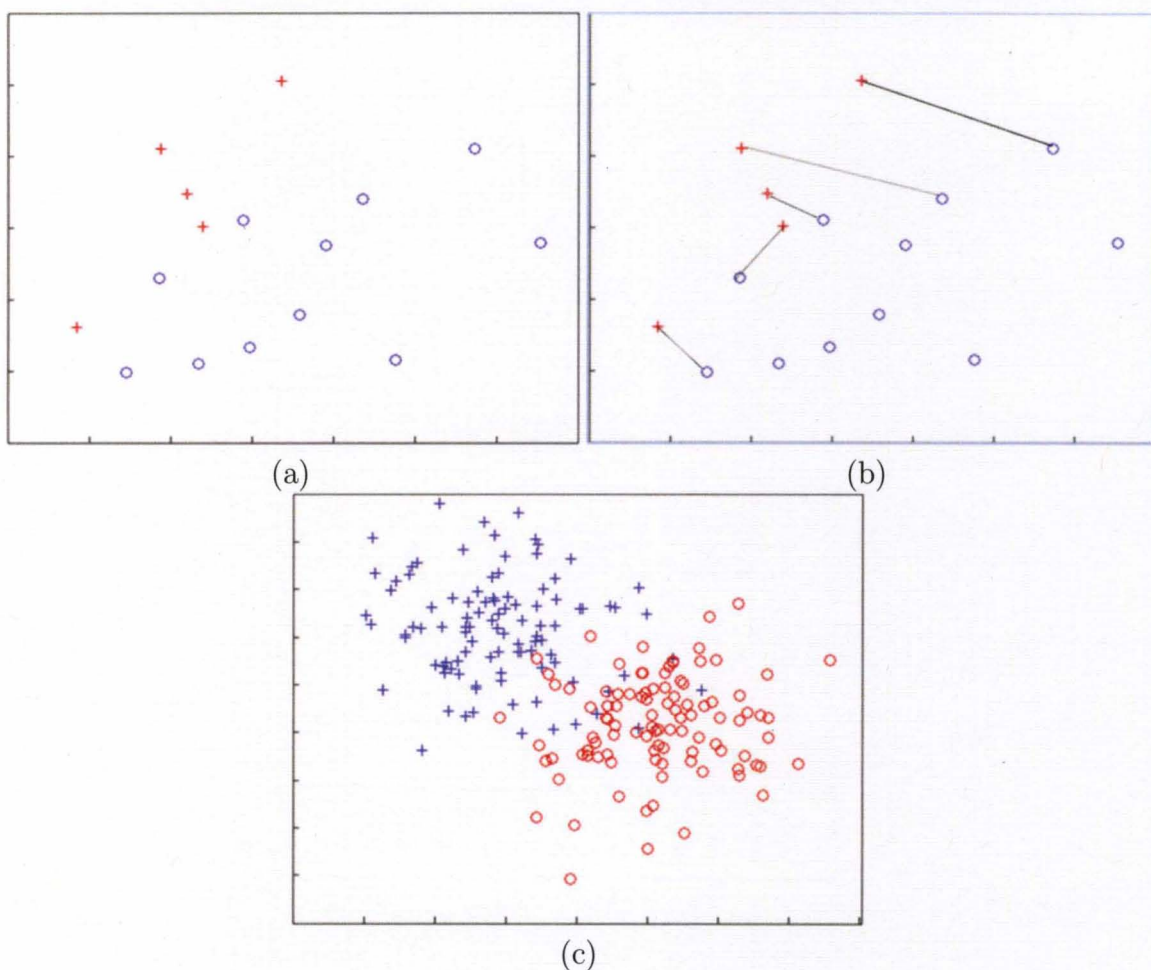


Figure 17. Demonstration of the use of pairwise constraints. (a) Enlarged view of the overlap in clusters from Figure 16(b). (b) Resulting constraints constructed from known information in Figure 16(a). (c) Reclustering with consideration to the constraints allows the algorithm to correctly partition the data set.

constructed using some form of prior knowledge, in our example we assume that we know that some of the '+'s belong to cluster 1 (blue), and some of the 'o's to cluster 2 (red). Figure 17(a) provides an enlarged view of the misclassification of points due to overlap. In Figure 17(b), constraints are constructed based on the known cluster assignments (see Figure 16(a)). Figure 17(c) displays the clustering results when the ACC was used with the constraints in Figure 17(b). As it can be seen, the use of 5 constraints has corrected the 16 misclassified samples.

The use of pairwise constraints implies a form of semi-supervision in the clus-

tering process. The use of class labels in the previous example is only one form of supervision information that can be applied in the form of constraints. In reality, any methodology that allows comparison between points can be used to construct constraints. In the following section, we present various methods of using contextual information to formulate pairwise constraints in the ACC algorithm.

C Constraint Selection

Constrained clustering is a recent area of research that has seen a steady amount of exploration [26, 27, 91, 99, 100]. Methods for applying constraints for image database categorization are usually application dependent and can be divided into three primary categories.

1. **Obstacle objects as constraints:** An obstacle can be defined as a physical object that obstructs the reachability among data objects. In a geographical setting these obstacles can be rivers, lakes, mountains, bridges, highways, etc. in an urban setting. Typically, the effects of these obstacles can be adverted by redefining the distance functions used among objects [38, 101, 102, 103].
2. **Feature-based constraints:** Methods using cluster-level constraints typically impose a significant number of constraints, using the same information available in the feature vectors, during clustering. Primarily, these constraints do not provide additional information and are implemented as a method to directly influence the inter- and intra- cluster distances. For instance, in [91, 104], the authors introduce constraints on the minimum and maximum separation of points within clusters, while in [105], the authors impose balancing constraints in attempts to avoid small or empty clusters.
3. **Side-information-based constraints:** These constraints are formulated be-

tween pairs of individual data objects using side information. In most cases, (e.g. [26, 82, 90, 91]), each pair of points defined by a constraint are implied to belong to either the same cluster or different clusters. Typically, these constraints are selected using metadata or some form of background knowledge.

In Chapter III, we proposed the ACC semi-supervised clustering algorithm. This approach uses side-information-based constraints. Particularly, these constraints provide suggestions on how pairs of points should or should not be grouped while not necessarily forcing their satisfaction.

In the remainder of this chapter we will present methods for creating constraints using various methods that can take advantage of available side information. In all cases, our algorithm’s primary role is to cluster based on visual content, using the features described in Section 1, and the information used for creating constraints comes from a contextual nature.

1 Active Selection of Constraints

Typically, an algorithm that uses partial supervision in the form of pair-wise constraints should include a strategy for selecting the constraints. In most cases, it is assumed that a various amount of information is known *a priori* [25, 91, 26]. In many cases, the information is provided in the form of class labels and constraints are selected at random. This method requires a minimal amount of system interaction. Other methods do not rely on random constraint selection [26, 83]. In these cases, the constraints are actively created using available information and some sort of external interaction from either the system or the user.

Coinciding with the decision of random versus active constraint selection, the overall number of constraints created is another important selection criteria. Typically, in the cases where a random selection method is used a fixed number of con-

straints are generated and introduced to the system in their entirety. For an active selection schema the constraints are typically created intermittently. This method allows for the algorithm to take an active learning approach to the clustering process.

The proposed ACC algorithm uses an active constraint selection strategy. First, the algorithm initializes the clustering process by making passes on the data without using constraints. This performs unsupervised clustering which attempts to discover underlying patterns in the data. Second, the algorithm begins the semi-supervised portion of the clustering process by defining a maximum number of constraints, Con_{max} , to create at each interval or epoch.

2 Exploring the Unsatisfied Should-not links to find the optimal number of clusters

The inclusion of partial supervision in the development of the ACC algorithm was intended to assist in the search for the optimal partition. Constructing pairwise constraints for use as suggestions towards cluster formation during the convergence process is one feature of the proposed ACC algorithm. Another feature is the use of pairwise constraints to assist with the search for the optimal number of clusters.

During the intermediate steps of the algorithm, a test is performed to check the satisfaction of the constraints. For example, if a *should-not* link constraint is violated, the system attempts to satisfy the constraint by splitting the cluster that contains the violation into two clusters. In particular, when a constraint is violated, each point defined by the constraint is used to create a new cluster, and the cluster count is incremented by one. Then, images are assigned to the new clusters based on their minimum distance.

Since the ACC uses methods from the CA [42], the number of clusters can be reduced, using competitive agglomeration. The constrained portion of the ACC

algorithm on the other hand uses the constraints to iteratively split and increase the number of the clusters. Once the cluster is split there is no guarantee that the agglomeration will not merge the clusters again, so the splitting function continually checks for satisfaction on each clustering iteration. Below is a summarization of the cluster splitting function:

Unsatisfied Should-not-link Cluster Splitting

For each (x_j, x_k) in the should-not-link constraint set, \mathcal{N} ;
 IF $\text{cluster}(x_j) = \text{cluster}(x_k) = i$;
 Calculate $t = \text{argmax}(d(x_j, c_i), d(x_k, c_i))$ using (47);
 Create new cluster, C_l , and let its center, c_l , be x_t
 End
End
 Reassign all points in cluster C_i to clusters C_i and C_l based on minimum distance;

D Categorization using the ACC with Constraints derived from Spatial Information

In recent years, the amount of information that is captured when a picture is taken has increased substantially. One area of interest is the inclusion of geographical location information in the form of latitude and longitude coordinates from the GPS standard [106, 107]. These images are referred to as geo-referenced images. Using this information it is possible to superimpose images on a map precisely where they were taken. Users are then able to visualize their photos highlighting their travels, or allow others to view images from places they plan to visit. The issue now becomes how to navigate through a large number of these images efficiently. In other words, how to avoid visualizing a map that is not completely inundated with overlapping images. Figure 18(a) shows an example of a map overwhelmed by images while (b) illustrates the same data set clustered so that the images can be navigated and viewed efficiently.

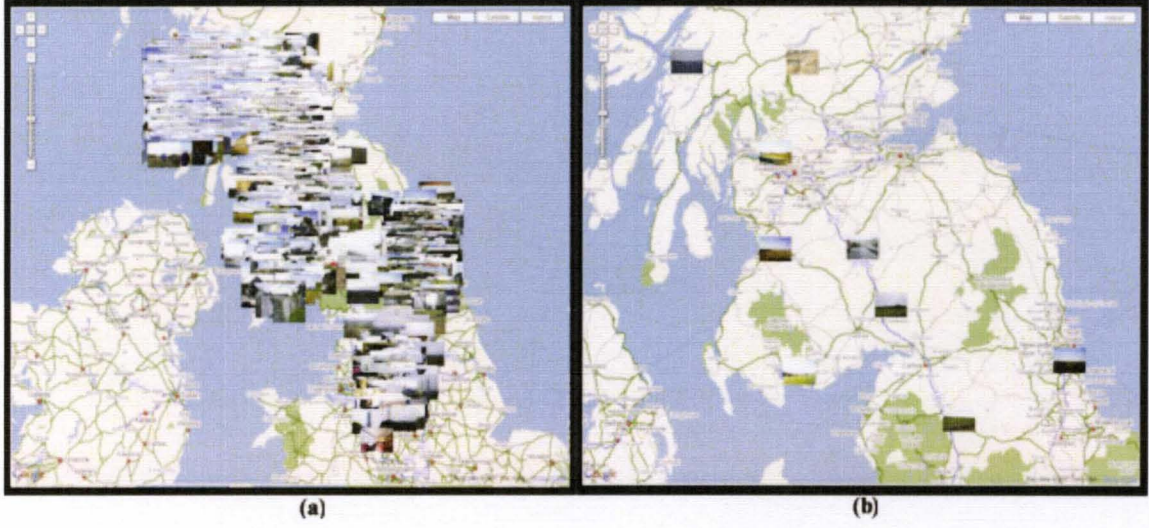


Figure 18. Comparison of a map cluttered with images versus a map with clusters of images.

1 Cost of Violating Spatial Constraints

In the proposed ACC algorithm, a cost, ρ_{jk} , is associated with each constraint. This cost is a weight that defines how strongly each selected *should-not* link constraint should be satisfied and is proportional to the spatial distance, θ_s (see Figure(19)). That is, during the constraint selection, the value of θ_s is dynamically selected to reflect the current resolution of the region being viewed. In other words, as the current region expands and retracts, the value of θ_s fluctuates proportionately. For our application, ρ_{jk} is computed using

$$\rho_{jk} = \begin{cases} \frac{d(\mathbf{x}_j, \mathbf{x}_k)}{3\theta_s} - \frac{1}{3} & \text{if } \theta_s < d(\mathbf{x}_j, \mathbf{x}_k) < 4\theta_s \\ 1 & \text{if } d(\mathbf{x}_j, \mathbf{x}_k) \geq 4\theta_s. \end{cases} \quad (67)$$

For the case of *should* link constraints, the cost κ_{jk} is calculated as $1 - \rho_{jk}$. These calculated costs return a normalized value retaining the notions of *distant* and *nearby* using the spatial contextual information.

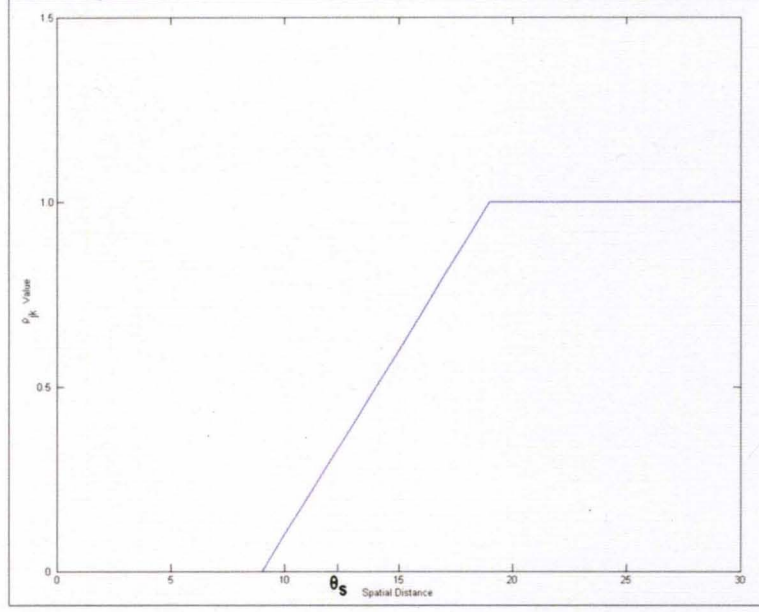


Figure 19. Calculation of the rho function.

2 Spatial Constraints Construction

Creating side-information-based spatial constraints can be as simple as using the spatial distance between pairs of images. If each image in a data set is tagged with spatial coordinates then the spatial distance between any two images ($\mathbf{x}_i, \mathbf{x}_j$) can be computed using

$$D_s(x_i, x_j) = \theta_E * \arccos(\sin(lat_{x_i}) * \sin(lat_{x_j}) + \cos(lat_{x_i}) * \cos(lat_{x_j}) * \cos(long_{x_j} - long_{x_i}))$$

where $\theta_E = 3959$ which is the earth's radius in miles, and $lat_{x_i} = (\text{latitude of } x_i)/(180/\pi)$, and $long_{x_j} = (\text{longitude of } x_j)/(180/\pi)$. Therefore, if $D_s(x_i, x_j) > \theta_s$, the pair of images (x_i, x_j) should be included in the set of *should-not* link constraints. We should note that with spatial information, we cannot create *should* link constraints based on spatial distances only considering the dependence on the visual content of the images. In other words, two images that are spatially close should not necessarily be assigned to the same cluster. Thus, we require that the images be similar (e.g.

assigned to the same cluster in an initial clustering step) and are spatially close in order to use them to create a *should* link constraint.

Figure 20 illustrates a subset of images in the spatial domain, where the images come from three distinct regions. Figure 21 shows the same subset of images in the feature domain, where the color of the outline represents the image’s respective region. We should note here that several images that are captured in different regions could be visually similar. Figure 22(a) shows the spatial layout of the images and Figure 22(b) represents the results of a typical clustering algorithm, where the colors and shape represent the two distinct clusters. As it can be seen, without including spatial constraints, we obtain two clusters of images and each cluster includes images from different regions. This clustering may not be useful for image navigation purposes. In Figure 22(c) we show few *should* link constraints given by dashed lines between images that are similar and spatially close, and *should-not* link constraints are solid lines between images of distinct spatial locations. Figure 22(d) displays the clustering results of the ACC that take the constraints into consideration. As it can be seen, the clusters found without constraints (displayed in Figure 22(a)) are now split into four clusters, in order to satisfy the constraints without affecting image content similarity.

3 Experimental Results using Geo-referenced Data

This section presents an application of the ACC algorithm involving an interactive and dynamic categorization of geo-referenced images. Our approach is illustrated using an example application with a collection of geo-referenced images. The image database is compiled from several distinct regions worldwide, and includes a collection of 2,023 geo-referenced images. Each of the photos have been automatically tagged with the capture (latitude and longitude) point by the digital camera. Photos were compiled from multiple distinct geographic locations: Kentucky, Mississippi, Atlantic

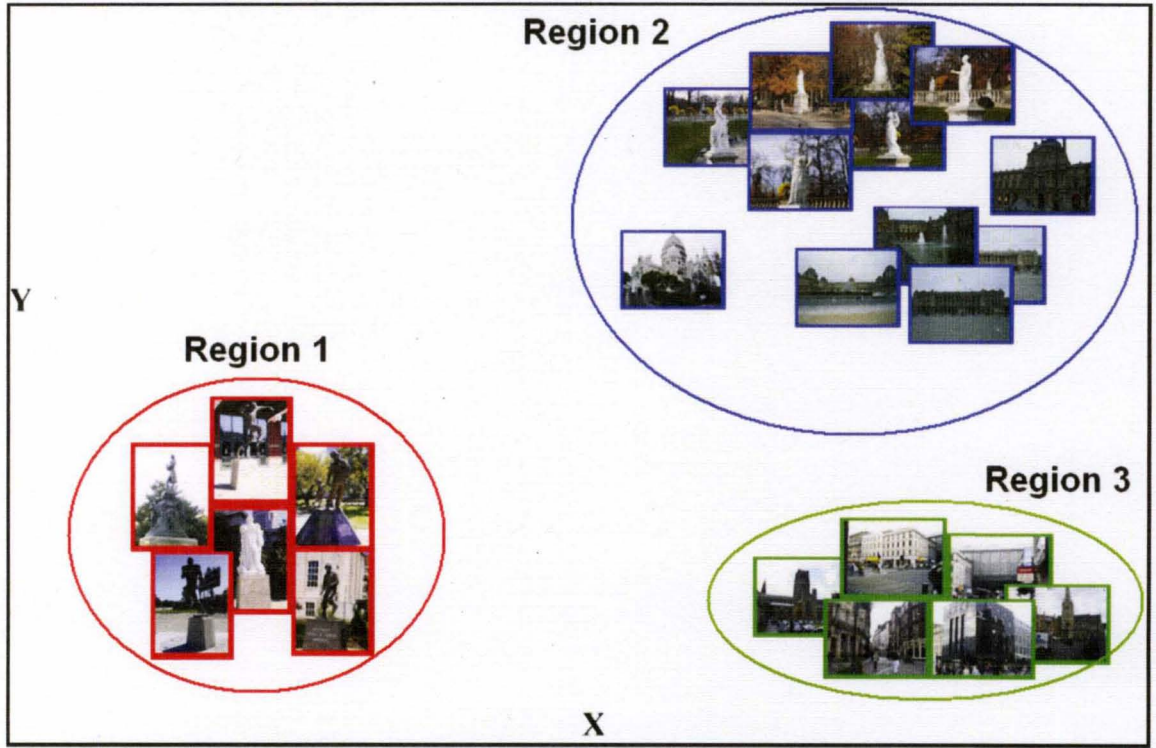


Figure 20. Spatial layout of a subset of images tagged with spatial coordinates. Each region corresponds to a distinct area across the globe.

City (USA), Paris (France), and Scotland (UK).

One of the main issues that the ACC attempts to alleviate by incorporating spatial constraints is the clustering of images whose content is similar, yet are separated by great distances. An example of this issue is shown in Table 6.

Each image is characterized by the features described in Section 1. For each feature subset, FS^s , the Euclidean distance is used as the distance function.

$$d(\mathbf{x}_j, \beta_i) = d_{ij} = \|\mathbf{x}_j - \mathbf{c}_i\|^2. \quad (68)$$

Considering each subset has a different number of dimensions and dynamic regions, each partial distance is scaled using

$$\tilde{d}_{ijs} = \frac{d_{ijs}}{\bar{d}_s}. \quad (69)$$

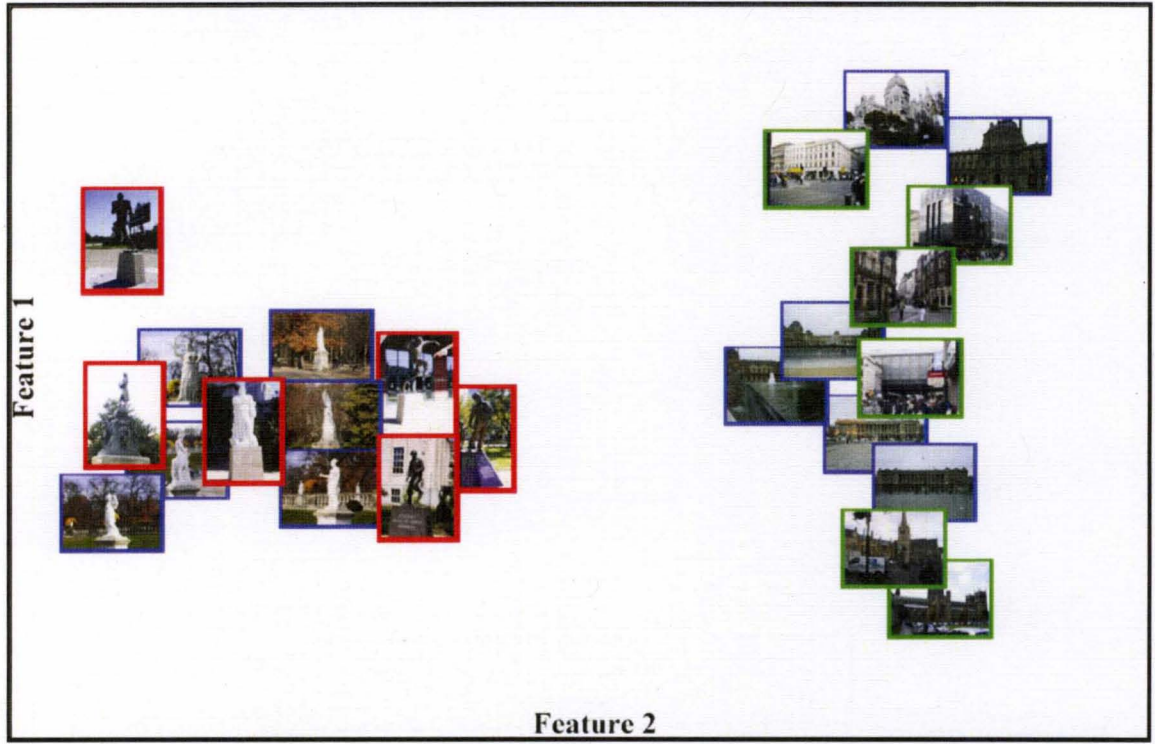


Figure 21. Layout of the images in Fig. 20 in the feature space. The image borders represent the three different geographical regions.

Where \bar{d}^s is an estimate of the average intra-cluster distances of FS^s , and is pre-computed using the FCM [40] for a small number of iterations using each feature set separately, and then using

$$\bar{d}_s = \frac{\sum_{i=1}^C \sum_{j=1}^N u_{ij}^m d_{ijs}}{\sum_{i=1}^C \sum_{j=1}^N u_{ij}^m}.$$

The parameters of the ACC algorithm were set as follows. The maximum number of clusters is set to $C_{max} = 50$, the fuzzifier set at $m = 1.5$, and the discrimination exponent is set to $q = 2.0$. The constraint importance factor γ is calculated using (37). For the agglomeration constant, α in (66), $\eta_0 = 2.0$ and $\tau = 20$. The spatial distance threshold, θ_S , during these tests is dynamic and is set to one fourth the largest possible spatial distance of the current view in the application.

Initially, the collection is clustered off-line into 20 clusters. A display of these

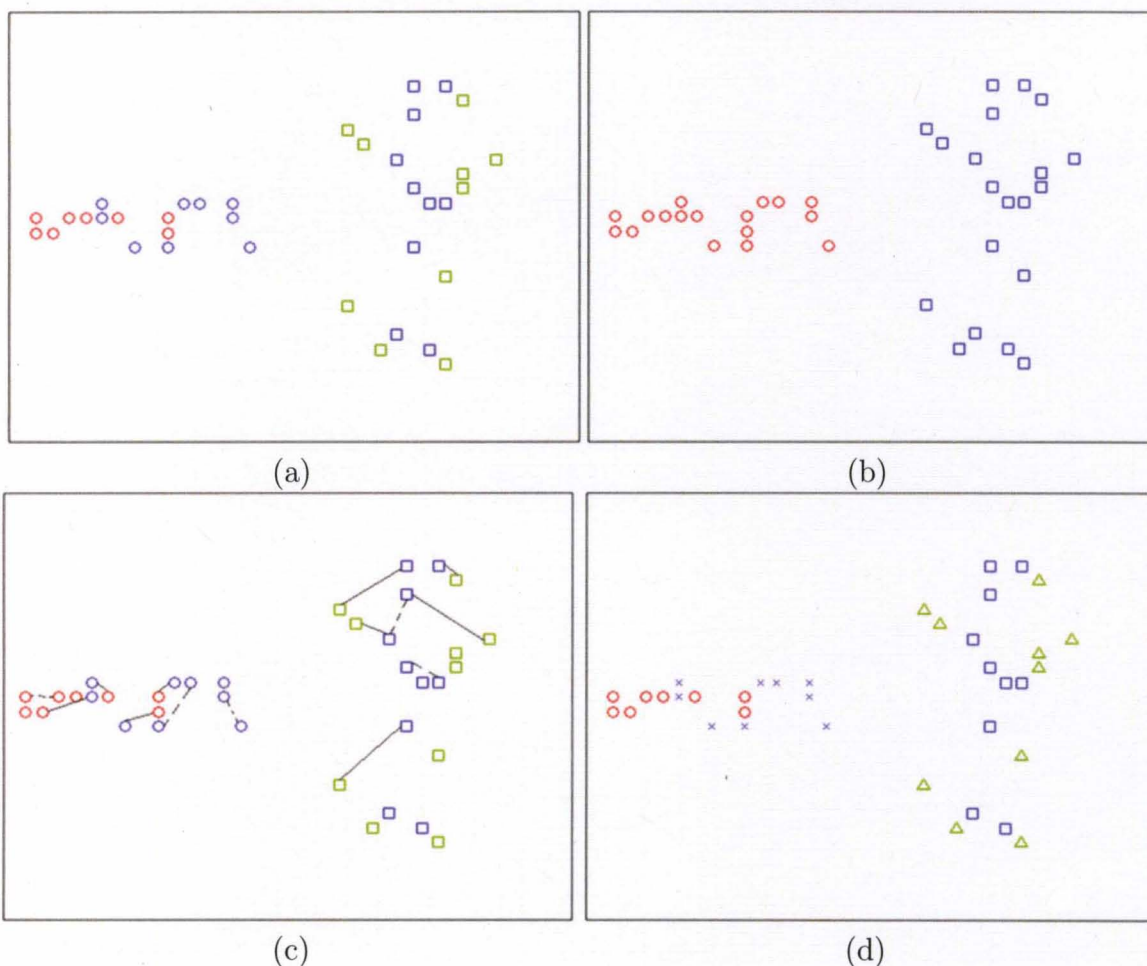














Figure 22. Application of spatial constraints. (a) An abstract view of the images in Fig 20. (b) Results generated by a typical unsupervised clustering algorithm. (c) A set of constraints are selected between points that are in different regions but should and should not be in the same cluster. (d) Partition generated with the consideration of the constraints where the number of clusters is expanded to four.

clusters is shown in Figure 23(a) where each point indicates the location of one cluster representative. In this view, θ_S is set to approximately 1500 miles. Therefore, no constraints are created between images within the US region (i.e. Kentucky, Mississippi, etc.) and likewise for the images in the European region, all constraints in this case are Trans-Atlantic. At this zoom level, the images are represented as markers in order to keep the map from being too cluttered. Next, Figure 23(b) shows the view after the user zoomed into the USA region represented by the box in Figure 23(a) which contains approximately 300 photos initially clustered into 4 clusters. Following

TABLE 6

Similar images from different geographical locations

Images			Cluster	Region
			Cluster 1	Region 1
			Cluster 1	Region 2
			Cluster 2	Region 3
			Cluster 2	Region 4

this action, the ACC algorithm is used to recluster the 300 images included in the selected region with stricter spatial constraints, i.e. smaller θ_S . The re-clustering process results in 12 new representative clusters. These representatives are now shown as images. In this zoomed view, constraints between the three US regions are present, where in the global view these images were not considered distant from one another.

Continuing to zoom on interesting areas, shown by the square in Figure 23(b), the new region contains about 70 images. Again, icons from the resulting re-categorization are shown in Figure 23(c). Zooming in one more time shows 12 small clusters, on the city level of the map, and spatial constraints are now created based on different regions of the city (Figure 23(d)). Finally, as the user zooms further, the enlarged region does not contain enough images to allow for re-clustering and all images are then shown for that region (Figure 25).

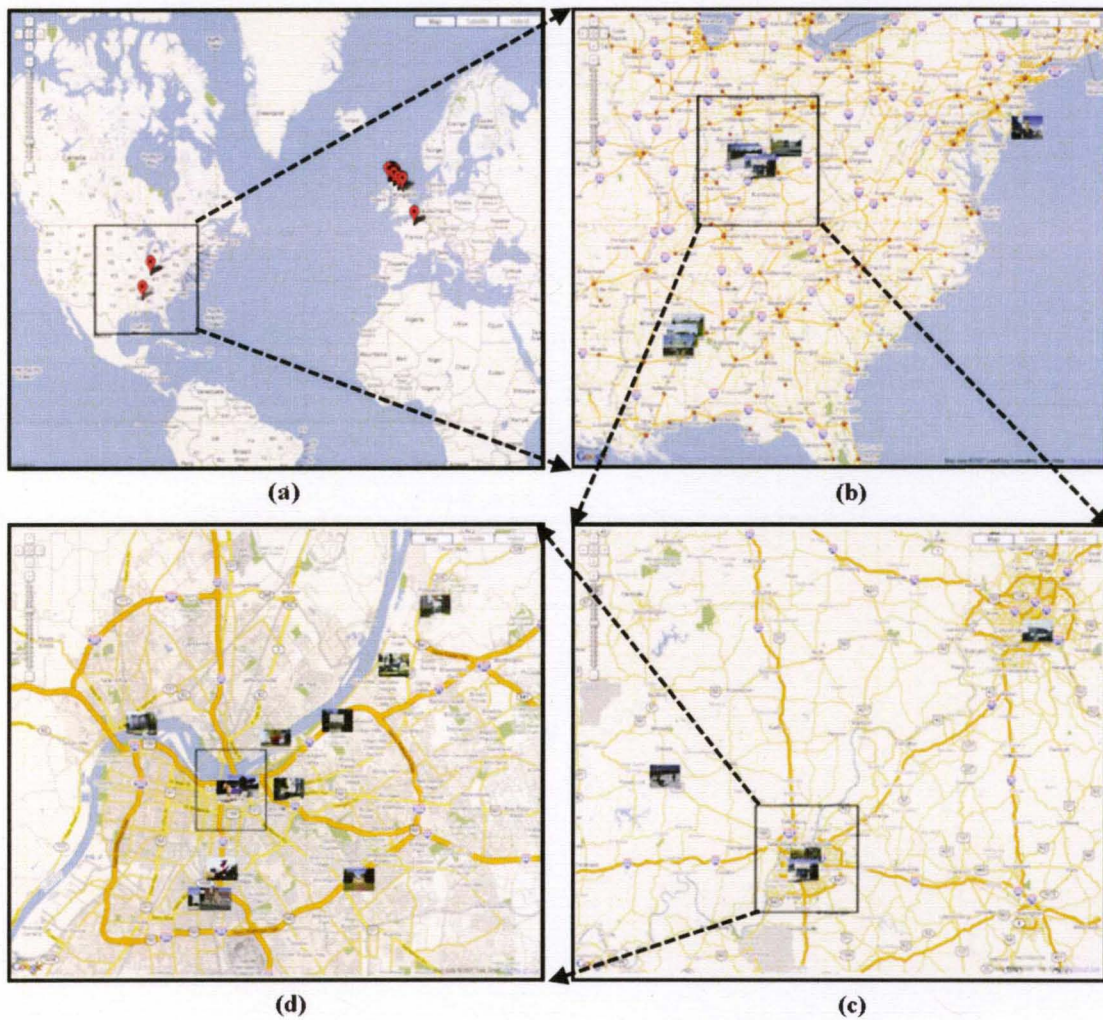


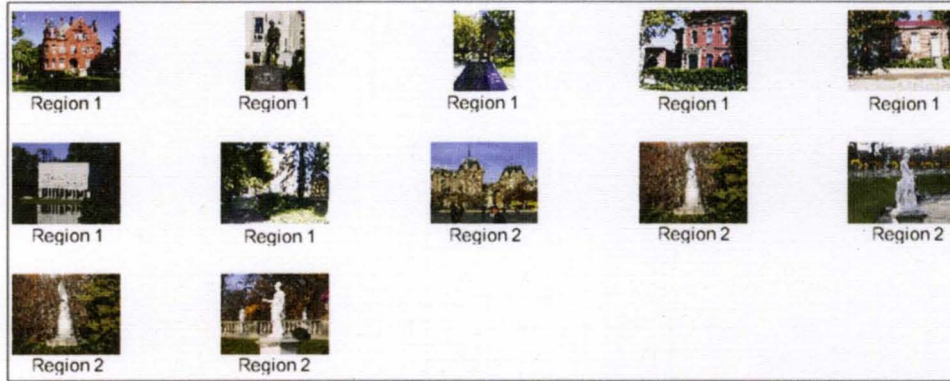
Figure 23. Illustrative example of spatial region expansion. (a) Overview of the image collection with representative clusters. (b) Reorganized data representing the USA region. (c) City level representation of data from selected region. (d) All images from street level region.

The images shown in Figure 25 are from clusters in the Louisville region and its surrounding area. In Figure 24, clusters resulting from unsupervised methods are displayed, where images from distinct regions are grouped into the same cluster. The results from the ACC algorithm show that previously combined regions have been extracted and images are now grouped not only based on their content features (Figures 25 and 26), but also using spatial context features creating hybrid results with visually similar images sharing similar regions.

E Categorization using the ACC with Constraints derived from Temporal Information

One of the strongest cues tied to memory is the aspect of time. Users intuitively associate "events" with the notion of time and content. This leads to organization of photos according to events for browsing, retrieval, and sharing tasks. Family vacations and functions are examples of events that are strongly tied to the notions of date and time. Unfortunately, events are still difficult to define in a consistent or quantitative fashion [108]. For example, simply trying to categorize multiple trips to the beach using low-level features is not a trivial task. This is because images could contain many different subjects such as the ocean, beach, or people and photos of the same scene could vary considerably depending on time of day or year [109].

With the wide availability of inexpensive "point-and-shoot" digital cameras which do not require the single use film rolls and photo development, the quantity of images being captured by the average user is rapidly growing. Although the use of film rolls is an aging technology, the typical user still retains a mental notion of photos taken in succession as being "from the same roll." Therefore the inclusion of temporal information to derive constraints could be useful in creating multiple clusters with visual similarities over varying time spans. This partitioning could simplify the tasks

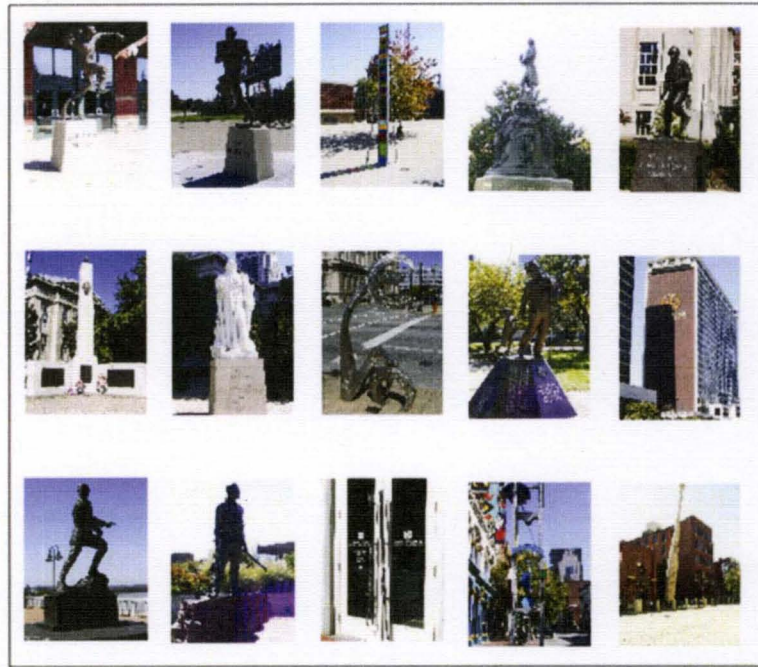


(a)



(b)

Figure 24. Unsupervised clustering results. (a) Results showing visually similar images from two distinct regions. (b) Results showing visually similar images from three distinct regions.



(a)



(b)

Figure 25. Viewing images contained in a cluster of interest. (a) Cluster containing images from Louisville. (b) Cluster containing images from surrounding Louisville area.

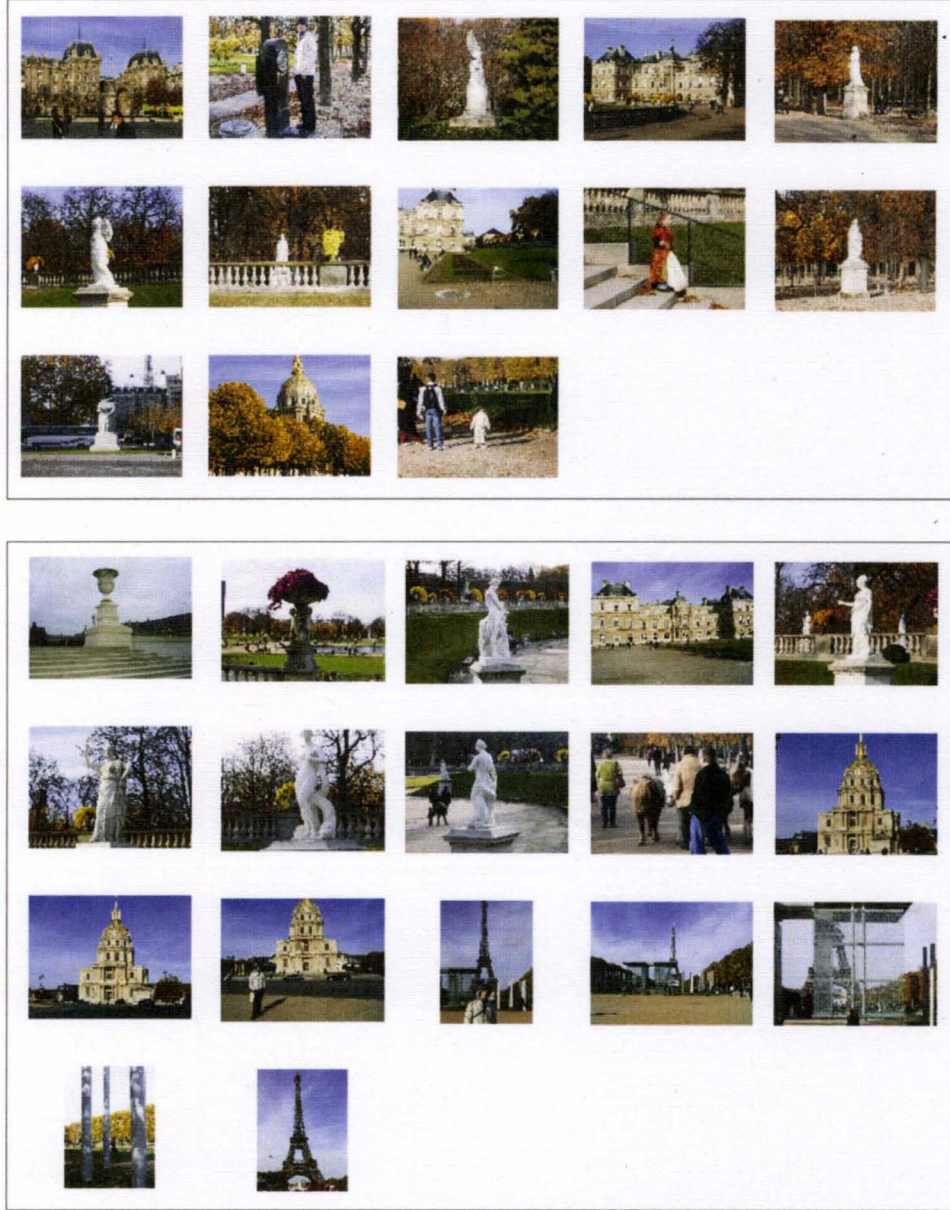


Figure 26. Viewing Cluster results from the Paris region.

of browsing and retrieving images in large image data sets.

1 Cost of Violating Temporal Constraints

Similar to the violation costs associated with spatial constraints (described in Section D.1), the cost of violating temporal constraints ρ_{jk} and κ_{jk} are proportional to the temporal distance θ_{time} . These costs should reflect the notions of *distant* and

nearby using the temporal contextual information. In particular, we define ρ_{jk} as

$$\rho_{jk} = \begin{cases} \frac{d(\mathbf{x}_j, \mathbf{x}_k)}{3\theta_{time}} - \frac{1}{3} & \text{if } \theta_{time} < d(\mathbf{x}_j, \mathbf{x}_k) < 4\theta_{time} \\ 1 & \text{if } d(\mathbf{x}_j, \mathbf{x}_k) \geq 4\theta_{time}, \end{cases} \quad (70)$$

and in the case of *should* link constraints, the cost κ_{jk} is calculated as $1 - \rho_{jk}$.

2 Temporal Constraints Construction

In constructing constraints using temporal information, it is necessary to quantify each date and time for comparative evaluation. In our approach, the difference between two images' temporal information, $(x_{i_{time}}, x_{j_{time}})$ is calculated as the total number of minutes between $x_{i_{time}}$ and $x_{j_{time}}$. That is, $D_{time}(x_i, x_j) = \|x_{i_{time}} - x_{j_{time}}\|$ in minutes. Given a threshold θ_{time} , if $D_{time}(x_i, x_j) > \theta_{time}$ the pair (x_i, x_j) should be included in the set of *should not* link constraints, and if $D_{time}(x_i, x_j) \leq \theta_{time}$ where x_i and x_j belong to the same cluster then (x_i, x_j) should be included in the set of *should* link constraints.

Assume that we have a collection of images, where each image is tagged with the date and time it was captured. Figure 27 illustrates a sample cluster based solely on the visual content of the images. In Figure 28, the contents of the cluster in Figure 27 are shown with respect to the temporal layout. Four distinct time spans are given, for example **time span 1** are images taken during a trip to a nature reserve in 2002, **time span 2** is from a family outing in 2004, **time span 3** contains images from a vacation in 2005, and images in **time span 4** are from a safari trip in 2008. An example of *should* and *should not* link constraints are given in Figure 29(a) and (b) respectively. Images in Figure 29(a) are selected as *should* link constraints due to their visual similarity and temporal proximity. Conversely, images in Figure 29(b) are chosen as *should not* link constraints because they are from different time spans. Figure 30 reveals the clustering results using both the image content and the selected



Figure 27. Sample cluster based on visual content.



Figure 28. Layout of Fig. 27 based on temporal information

constraints where the previous cluster in Figure 27 is now split into various clusters of visual similarity while respecting the temporal layout of the data set.



Figure 29. Application of temporal constraints. (a) A set of should link constraints between images with like time signatures. (b) A set of should not link constraints between images with differing time signatures.

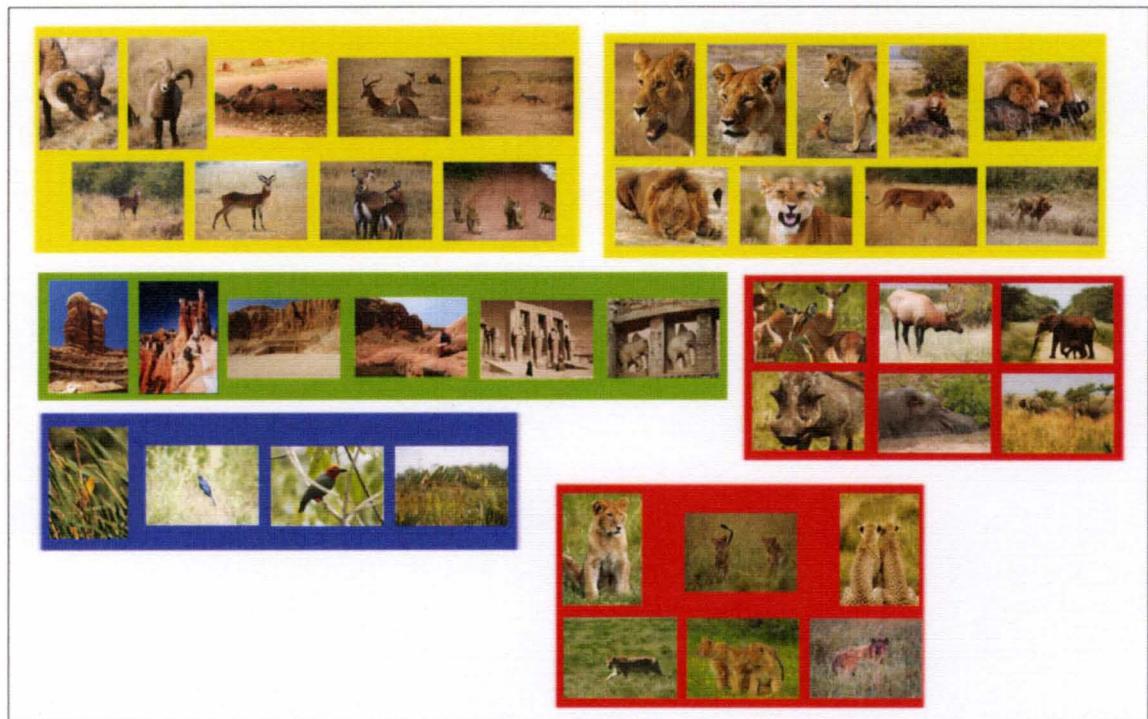


Figure 30. Sample clusters generated by the ACC algorithm using the temporal constraints in Fig. 29

3 Categorization using the ACC with Constraints derived from Textual Information

With the emergence of social networking sites which allow users to share pictures [2, 3] the number of images that contain some form of annotation is expanding. Although these annotations might not be extensive enough to allow for predefining categories used in supervised clustering, the inclusion of partial annotations is a natural constraint selection method for semi-supervised clustering [99, 100].

Using text as a feature in image classification [4, 110] requires feature extraction to use for comparison purposes. Suppose we have a set of images, I , with an associated set of textual keywords T . Comparisons made using all keywords may provide ambiguous results due to the frequency of certain keywords. The term frequency, inverse document frequency (TF-IDF) measure [111] is an accepted method for finding a set of *meaningful* keywords from a set of documents. Let t_i be a word used to annotate image i . The *term frequency* (TF) is defined as

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \quad (71)$$

where $n_{i,j}$ is the number of occurrences of the term t_i in image i and $\sum_k n_{k,j}$ is the sum of the occurrences of all terms in all images. The *inverse document frequency* (IDF) of term i is a measure of the importance of this term. The IDF is defined as

$$idf_i = \frac{\log \|D\|}{\|\{d : t_i \in d\}\|} \quad (72)$$

where $\|D\|$ is the total number of images and $\|\{d : t_i \in d\}\|$ is the number of images that contain the term t_i . Then, the TFIDF value for the term t_i in image i is defined as

$$TFIDF_{i,j} = tf_{i,j} \times idf_i. \quad (73)$$

A high TFIDF weight implies a high frequency term in a given image and a low image frequency of the term in the image collection. Using the TFIDF values, it is possible to

find the top k meaningful keywords in T denoted as T_k . Then using T_k , k -dimensional feature vectors are created for each image i where $\mathbf{KW}_i = [kw_1 \dots kw_k]$ and

$$\begin{cases} kw_i = 1, & t_i \in T_k \\ kw_i = 0 & \text{otherwise.} \end{cases} \quad (74)$$

4 Cost of Violating Textual Constraints

The cost of violating *should* link constraints, κ_{jk} , with respect to textual information can be calculated using the keywords associated with each image. Let $\mathbf{KW}_i = [kw_1 \dots kw_k]$ be a set of keywords associated with image i and $\mathbf{KW}_j = [kw_1 \dots kw_k]$ be a set of keywords associated with image j . Then, we define κ_{jk} as

$$\kappa_{ij} = \frac{\|K_i \cup K_j\| - \|K_i \cap K_j\|}{\|K_i \cup K_j\|}. \quad (75)$$

In other words, the cost of violating a *should* link constraint is the ratio of the difference in the sizes of the union and intersection of the two keyword sets, and the size of the union. For *should not* link violations, ρ_{jk} is set to one, because all *should not* link constraints share no words.

5 Textual Constraints Construction

The similarity between two images (x_i, x_j) , each with a respective set of keywords KW_i, KW_j is defined as

$$S_{text}(x_i, x_j) = \frac{(\|KW_i \cap KW_j\|)}{(\|KW_i\| + \|KW_j\|)}, \quad (76)$$

or the ratio of words shared by the images x_i and x_j to the total number of words between these images. In (76), $S_{text}(x_i, x_j) \in [0, 1]$ represents the percentage of words shared by the image pair. In the case of *should not* link constraints, if the pair (x_i, x_j) have no words in common, or $S_{text}(x_i, x_j) = 0$, then (x_i, x_j) should belong to the

set of *should not* link constraints. On the other hand, if $S_{text}(x_i, x_j) \geq \theta_{text}$, where $\theta_{text} \in [0, 1]$ is a threshold, then (x_i, x_j) belongs to the set of *should*-link constraints.

The following illustrative example demonstrates how creating textual constraints can lead to clustering results with improved cluster semantics. Figure 31 contains a subset of images with varying levels of visual similarity. Overall, the images are taken either late in the evening or at night. Therefore, they all are dark images with a strong central object. This leads to clustering results that rely more on color content than texture content. Figures 32(a)-(d) illustrate the clusters based solely on visual content. The first cluster in Figure 32(a) contains sunsets without a strong presence of a setting sun where Figure 32(b) and (c) consist of sunsets and night images where the sun is visible with prominence in the colors orange and red respectively. The last cluster, Figure 32(d), shows images of people taken at night.

Next, *should* link constraints are constructed from image pairs if they share at least one word. *Should not* link constraints are constructed between images that share no words. Figure 33(a) displays the identified *should* link pairs, and Figure 33(b) displays the identified *should not* link pairs. Figure 34 displays a sample cluster with the consideration of the chosen constraints. As expected, clusters generated with constraints group all the images of sunsets even though some images show more dominance in red or orange, and the sun is visible in a number of images but not in others. This simple example illustrates how textual constraints can provide the ability to add another level of semantics to the clustering results.

F Experimental Evaluation and Comparison

The evaluation of clustering results is a difficult process. Humans, through experiences and knowledge gained over their lives, can easily place similar images into groups based on the content of the images. Having a computer create similar

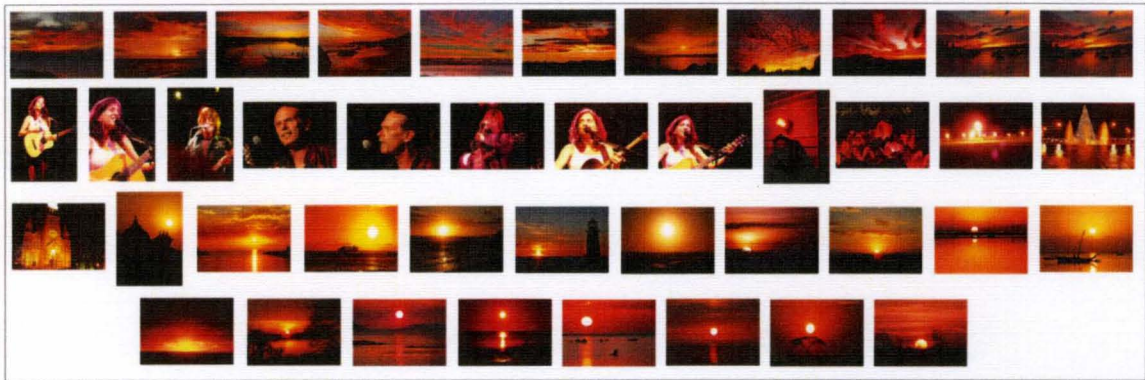


Figure 31. Sample data set for textually constrained clustering.

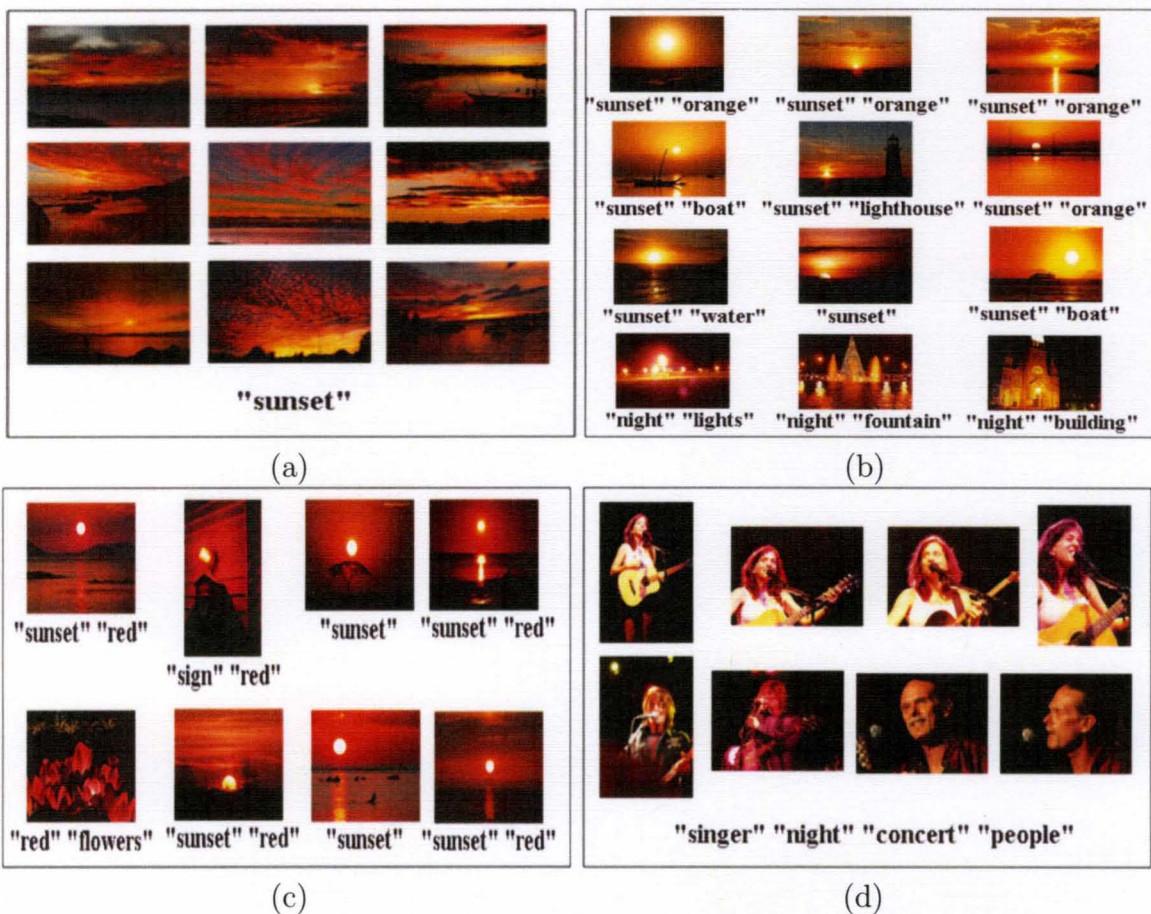


Figure 32. Sample clustering of images in Fig. 31 clustered using image content. (a) Images of sunsets with the sun under the horizon. All images tagged with the keyword "sunset". (b) Various images with a prominent orange coloring and associated image tags. (c) Cluster containing red images with assigned annotations. (d) Partition containing nighttime images of people.



Figure 33. Application of textual constraints. (a) A set of should link constraints between images that share the keyword "sunset". (b) A set of should not link constraints between images with no words in common.



Figure 34. Sample clustering displaying enhanced semantics from textual constraints.

groups based on the content of the images, on the other hand, is a challenging task.

The results are presented using two methods of validation. Both methods are comparative and show the performance of the ACC algorithm with respect to the performance of the SCADCA algorithm (see Chapter II.§C), and the PCCA algorithm (see Chapter II.§D.2). In Section 3, the results are evaluated objectively using known associations to obtain a validity score. Section 4 utilizes the human

element to validate the results of the ACC algorithm through subjective evaluation.

1 Image Collection

The evaluation techniques described in this section both utilize a sample data set consisting of 450 images. The features of each image are represented using the CSD feature subset with 128 color dimensions and the EHD feature subset (refer to Section 1 of this chapter). Each image is also annotated with a set of keywords varying in size from one to five semantically relevant words. Images were selected as part of this data set such that a few generalized themes were present, but the images within each theme vary by representation of their low-level features. The ACC algorithm attempts to alleviate the combination and separation of images with similar contextual information but differing content information. An example subset of images from this data set, demonstrating this issue, is presented in Table 7.













For all experiments, the parameters of the ACC algorithm were set as follows. The fuzzifier is set to $m = 1.25$, and the discrimination exponent is set to $q = 2.0$. The constraint importance factor γ is calculated using (65). For the agglomeration constant, α in (66), $\eta_0 = 3.0$ and $\tau = 80$. The distance calculations for these experiments are calculated using equations (68) and (69).

2 Incorporating Constraints

The constraints used in these experiments are gathered from image annotations associated with each image. The process of choosing textual based constraints is outlined in Section C.3 of this chapter. In our experiments the constraints are pre-computed on the basis that if a pair of images (x_i, x_j) share keywords they are included in the set of *should* link constraints, $(x_i, x_j) \in S$. Conversely if they share no keywords then they belong to the set of *should not* link constraints, $(x_i, x_j) \in N$.

TABLE 7

Images with similar contextual semantics and differing content information

Images			
Keywords	"sunset"	"sunset" "orange"	"sunset" "sun" "red"
Images			
Keywords	"beach" "ocean"	"beach" "sand" "ocean"	"beach" "ocean" "mountain"
Images			
Keywords	"building" "church"	"building" "city"	"building" "city" "night"
Images			
Keywords	"park" "tree"	"park" "lake" "grass" "tree"	"park" "grass" "rock"

For our experimental data set $\|S\| = 75000$ and $\|N\| = 5067$. These constraints are provided to the ACC as complete sets due to ensure that both the ACC and the PCCA algorithms utilize the same set of constraints.

3 Objective Evaluation

For this experiment set, the ACC algorithm was tested against the SCADCA and the PCCA algorithms with the results validated objectively. It is assumed that for the given data set, the ground truth, GT , is known. Then, the validity of a given cluster c_i , $p(c_i)$, is defined as the purity of the cluster, that is, the ratio of points that share the same ground truth value, to the size of that cluster. The purity of the resulting partition is the sum of each cluster's purity. In other words, we define the

purity of cluster c_i containing n points as

$$p(c_i) = \frac{1}{n} \left(\sum_{j=1}^n \sum_{k=j+1}^n f(x_j, x_k) \right), \quad (77)$$

where

$$f(x_i, x_j) = \begin{cases} 1 & GT(x_i) = GT(x_j), \\ 0 & otherwise. \end{cases}$$

The overall validity of the returned partition containing C clusters is computed using

$$\bar{P} = \frac{1}{C} \sum_{i=1}^C p(c_i) \quad (78)$$

where $0 \leq \bar{P} \leq 1$.

In our experiments, the value of C_{max} was set to 100, and the ACC algorithm discovered 28 clusters. The validity values obtained from each clustering algorithm is given in Figure 35. Figure 36 compares the number of satisfied constraints from the ACC and PCCA algorithms. As it can be seen, in both cases many of the *should not* link constraints were satisfied. We should mention here that the constraints used in both algorithms are soft constraints, meaning that the satisfaction of constraints is not forced, and unsatisfied constraints may be present. There is a noticeable difference in the satisfaction of *should* link constraints between the two algorithms. The definition of *should* link constraints implies that satisfaction requires all points to reside in the same cluster, but does not consider multiple clusters of like images. Therefore, one large cluster may contain many images that should be linked together and satisfy many constraints, but the overall validity of that cluster and the results may suffer. Therefore, even though the PCCA algorithm was able to satisfy more constraints than the ACC algorithm, the ACC algorithm provides a more valid partition.

Using the sample data set in Section F.1, Figure 37(a) illustrates the average objective function values and the variation per iteration computed using the SCADCA

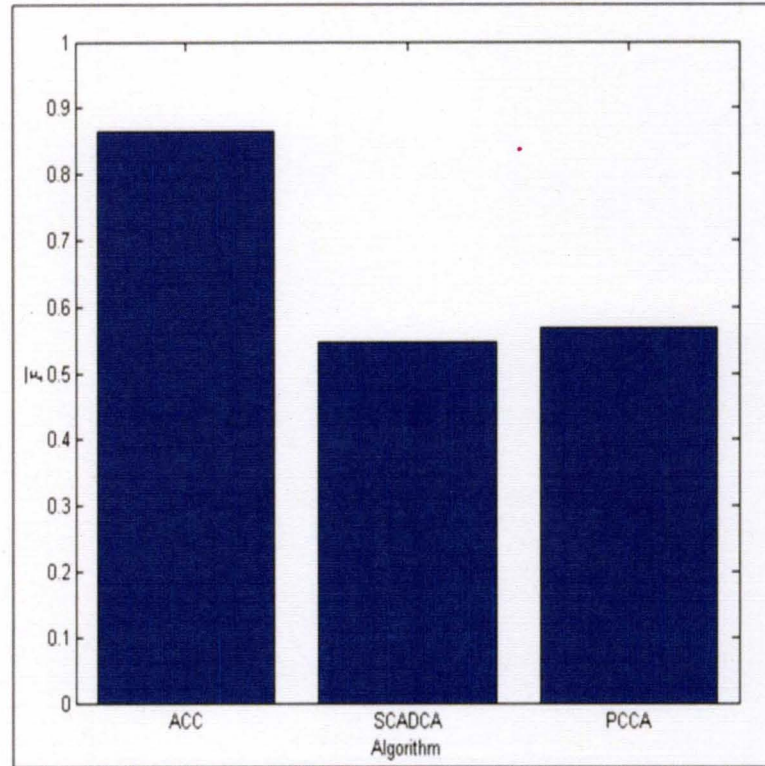


Figure 35. Average cluster purity of the partitions generated by the ACC, SCADCA, and PCCA algorithms.

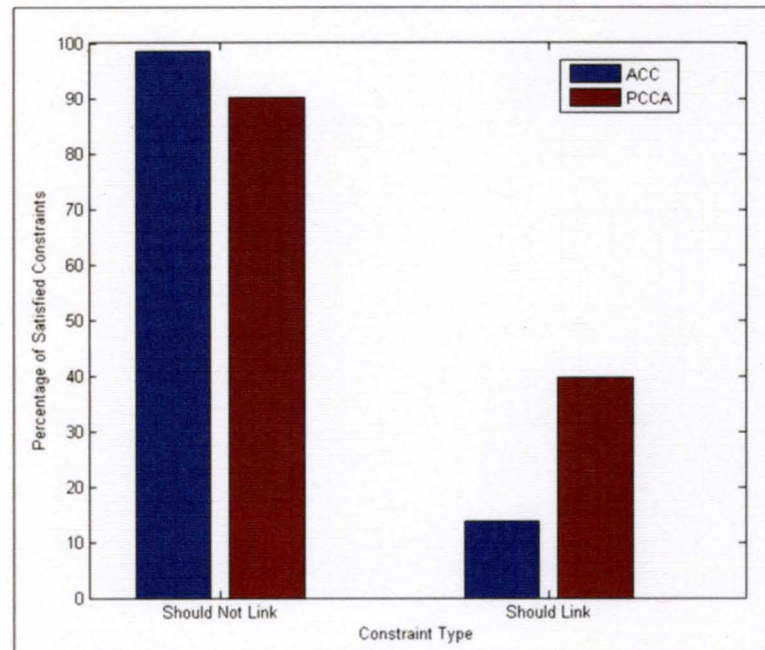


Figure 36. Percentage of satisfied constraints from objective experiments.

algorithm. As it can be seen, the SCADCA converged on average in 75 iterations and displays a small amount of variation in both the objective function and the standard deviation. This is primarily due to the SCADCA algorithm's early agglomeration properties in which the number of clusters does not drastically change throughout the clustering process. Next, Figure 37(b) shows the resulting objective function using the PCCA algorithm. As described in Chapter III.§E, if the algorithm in question has not converged after 100 iterations the algorithm is set to terminate. The PCCA fluctuations observed in the objective function again validate the assumption that the PCCA algorithm tends to become trapped in local minima. Figure 37(c) is the average values for the ACC algorithm. During iterations 40 - 60, the ACC algorithm suffers from effects of a local minima but unlike the PCCA, it is able to recover from these fluctuations and converge on average in 82 iterations. These convergence properties are reflected in the average run times given in Table 9.

TABLE 8

The observed running time for each algorithm on sample data set in Section F.1.

Algorithm	Run Time
ACC	2981.11s
SCADCA	455.20s
PCCA	4363.54s

The results in Table 9 indicate a substantial increase in the run time of the ACC algorithm with respect to the increase of the complexity parameters (see Chapter III.§D). In a dynamic environment, it is important to balance the time complexity and performance of the given algorithm. In Figure 38, we presented an example of dynamic image database categorization under spatial constraints using the ACC algorithm. The dataset in question contains approximately 10,000 images and using the ACC algorithm, with the active constraint selection paradigm, is estimated to perform the

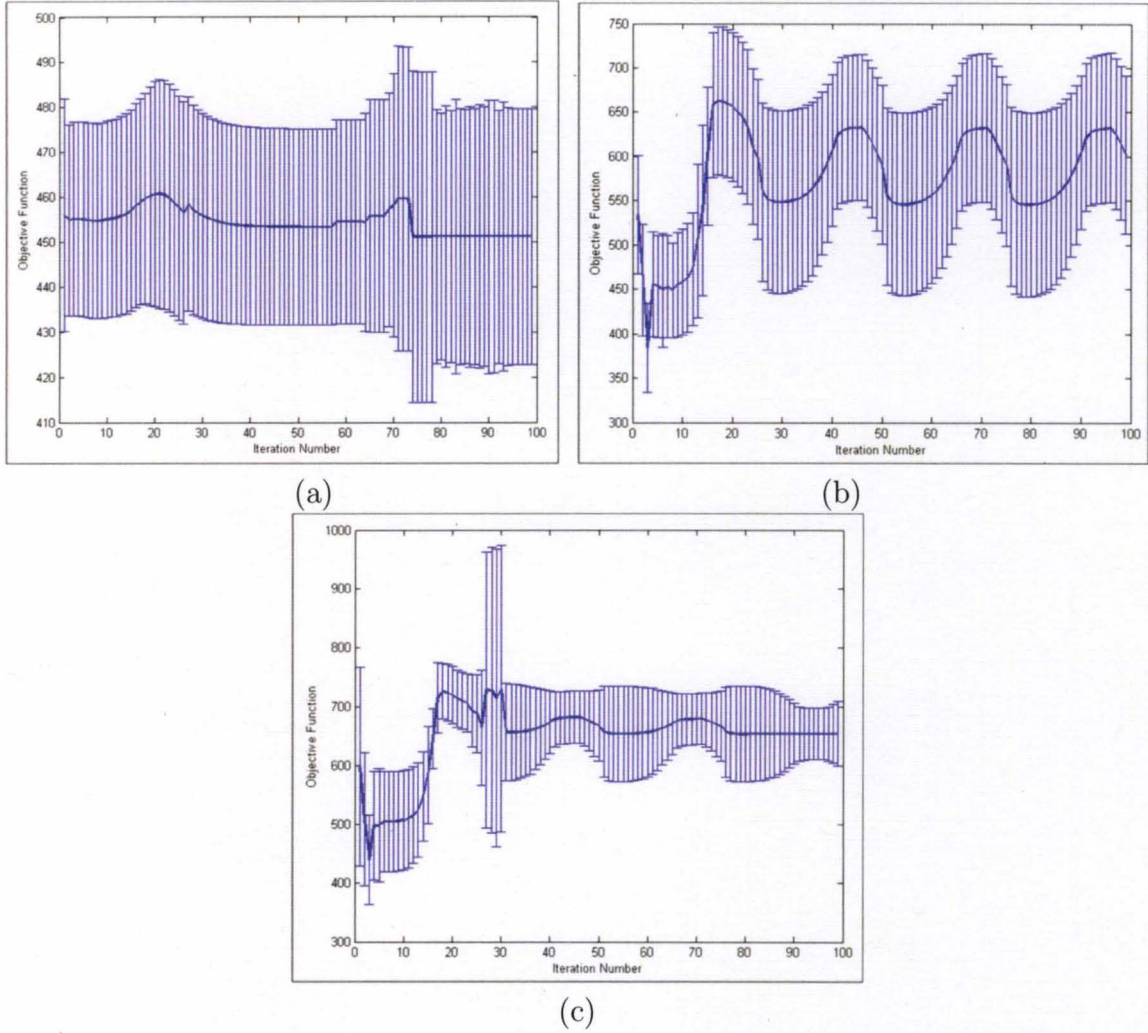


Figure 37. Evaluation of the objective function over 100 sample trials. (a) Results for the SCADCA algorithm (b) Results for the PCCA algorithm (c) Results for the ACC algorithm

initial clustering in approximately 6 hours. Therefore, in a dynamic application the initial clustering of the dataset is performed off-line. The user is first presented with the results of the initial clustering in Figure 38(a), and selects a region of interest. Contained in the selected region are approximately 1000 images, and are reclustered in under three minutes. The results of this reorganization is then presented to the user in Figure 38(b), and the user then selected another region of interest. The number of images in this region is now on the order of a few hundred and the ACC algorithm is able to reorganize this subset in approximately one minute. After selecting the final

area of interest, the new image subset does not require reorganization by the ACC algorithm due to the contained number of images and the subset is displayed in its entirety. A summary of the sizes and times associated with the off-line and online steps of the ACC algorithm in a dynamic application are provided in Table ??.

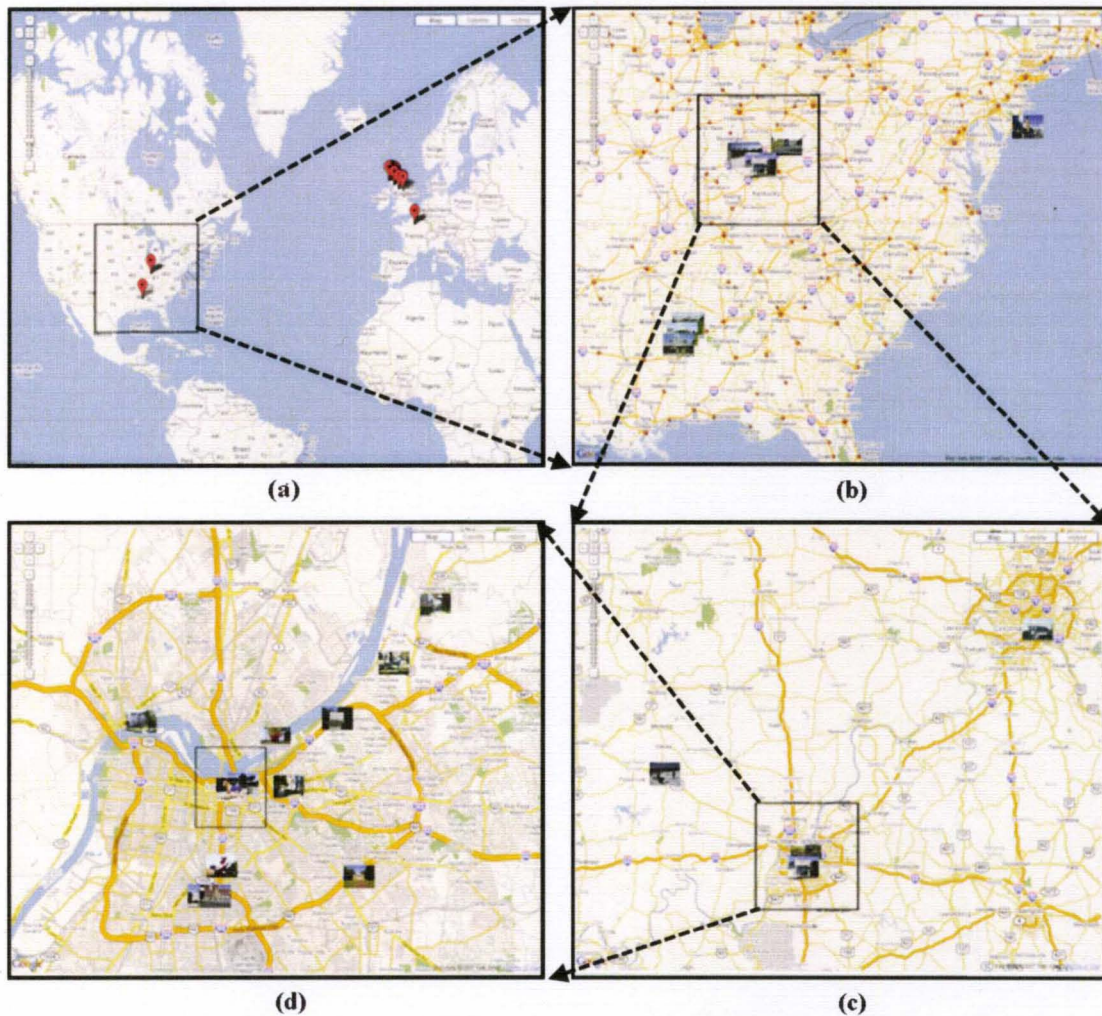


Figure 38. Illustrative example of spatial region expansion. (a) Overview of the image collection with representative clusters initially clustered in an off-line fashion. (b) Reorganized data representing the user's preference, processed online. (c) The images contained in the area of interest are dynamically reclustered with respect to the current region. (d) At the street level, the number of images contained the region do not require categorization and are displayed in their entirety.

TABLE 9

The observed running time of an example application of dynamic image categorization using the ACC algorithm.

Processing Method	Data Size	Run Time
Off-line	10000	340.2 min.
Online	987	2.78 min.
Online	241	0.93 min.

The following figures provide illustrative examples of clusters from the various partitions created during the objective evaluations. In Figure 39, we compare clusters from the ACC and SCADCA algorithm. With respect to the SCADCA results, we note the presence of green landscapes within the beach scenes. In this case, the use of partial supervision prevents the ACC algorithm from grouping these points. Figures 40 and Figure 41 compare results from the ACC and the PCCA algorithms. The point of interest is the variation in the quality of the results. In Figure 40 the result of the ACC algorithm are only marginally better than those of the PCCA algorithm. On the other hand, the comparison in Figure 41 illustrates a drastic improvement in the purity of the sample cluster. This trend continues in subsequent examples (Figure 42) with the purity of the clusters returned by the ACC algorithm displaying noticeably better clusters than the comparative algorithms. In most cases, the effects of the partial supervision had been evident in the case of *should-not* link constraints. In Figure 43, the results indicate the presence of *should* link constraints, where images contained in two clusters from the SCADCA partition (Figure 43(a-b)) are combined in the ACC partition.

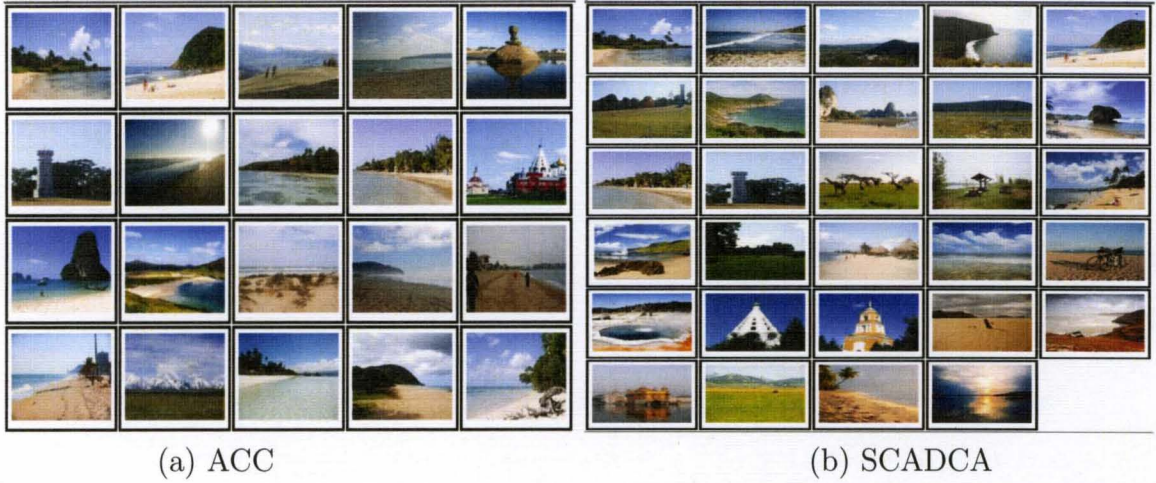


Figure 39. Comparison of clusters from the ACC and SCADCA algorithm. These results illustrate the benefit of using partial supervision.

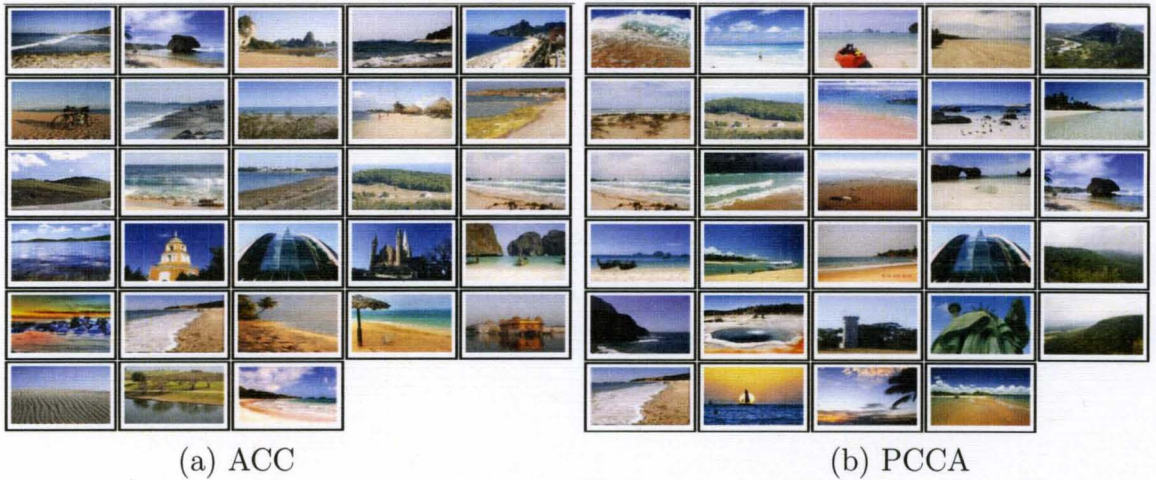


Figure 40. Comparison of ACC and PCCA clusters illustrating the benefits of using relevance feature weights.

4 Subjective Evaluation

From the inception of image clustering, the goal has been to create concise, meaningful clusters that assist users with retrieving and navigating large image collections. A persistent issue has been coined the semantic gap. This issue is present both in the creation of algorithms and their validation. In regards to validation, the performance of any given algorithm is subjective not only to the purpose, procedure, and design but also to those that view the results of the algorithm.

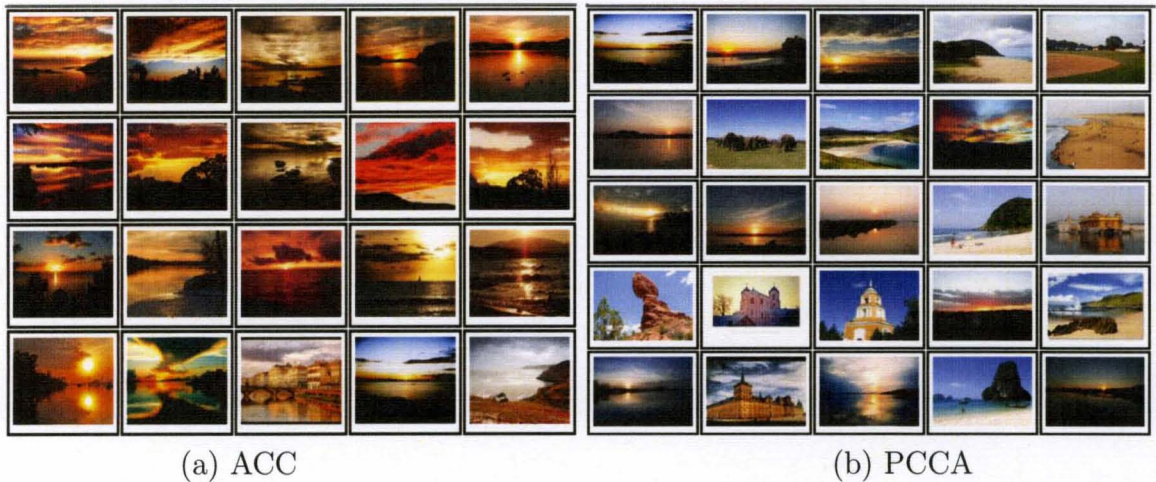


Figure 41. Results comparing the ACC and the PCCA algorithm, demonstrating increased clustering performance.

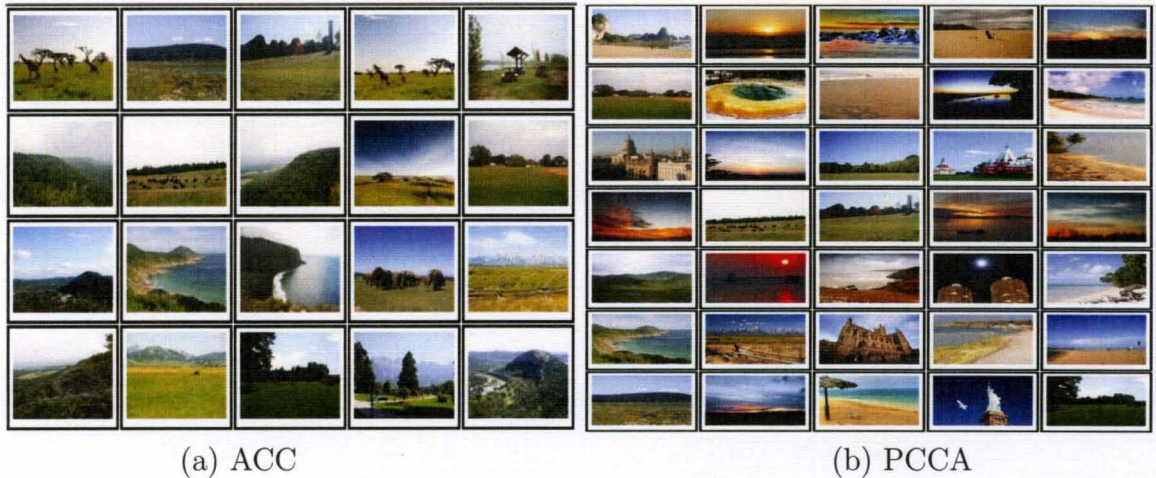


Figure 42. Comparing the ACC and the PCCA algorithm, where the increased clustering performance can be attributed to increased constraint satisfaction.

Figure 44 provides an illustrative example that arises when one considers the subjectivity between users under validation. One user might see these results and find three small circular clusters, while another users sees only two clusters. Although, a third user might again find two clusters their interpretation of the results is again different than the other two users. In Section 3, we objectively validated our results with respect to the semantic gap, in this section a subjective evaluation is presented in order to validate both issues of semantics and subjectivity.

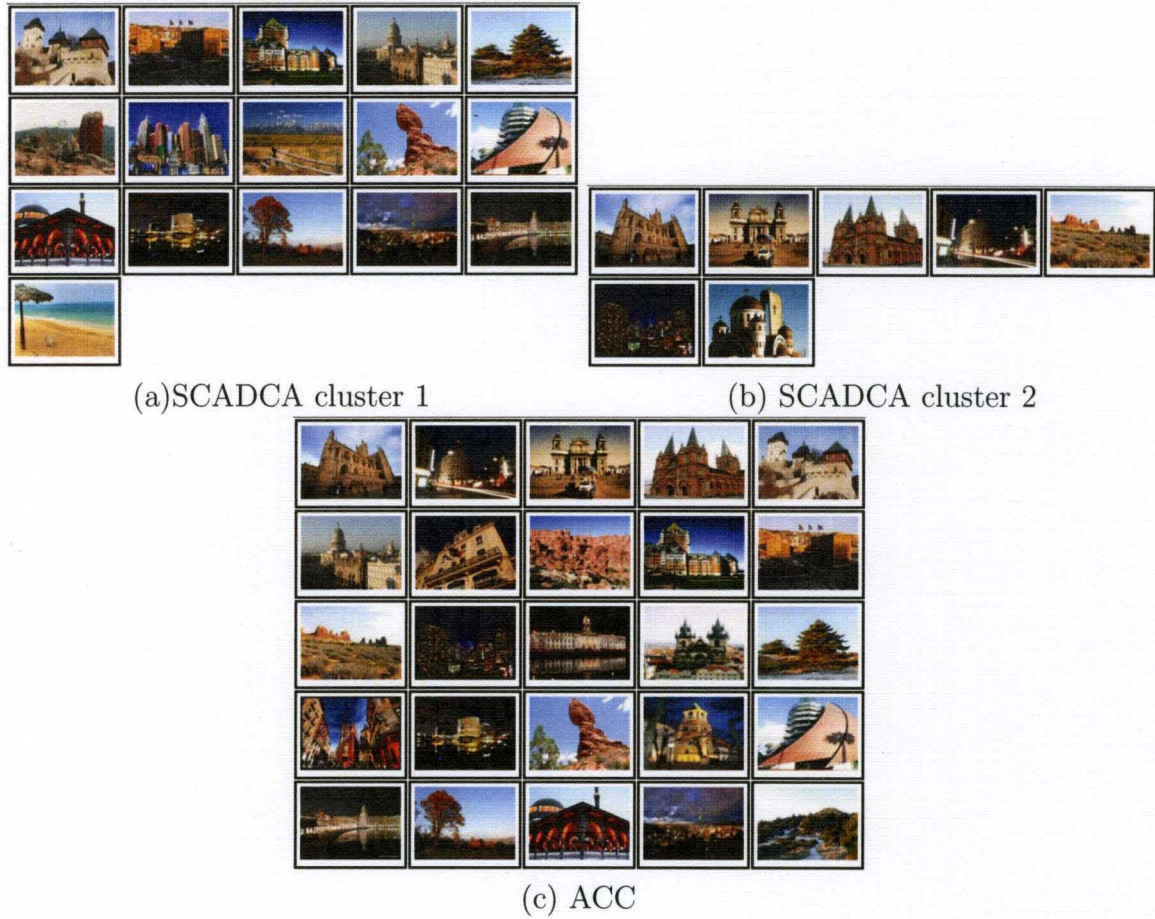


Figure 43. Results from the ACC algorithm indicating the presence of *should* link constraints.

For this set of experiments C_{max} was set to 50, and the results of the algorithms return a total of nine clusters, $C = 9$. Again, the ACC algorithm is validated against the SCADCA and PCCA algorithms as well as a random partition of the data set. Figure 45 shows a screen shot of the test given for subjective evaluation. With four different clustering methods and nine clusters, each user is presented with 36 result sets displayed at random. The user is then asked to provide their subjective opinions and select only the images that are relevant to the given cluster.

The results of this test return a set of values for each cluster that represent the cluster's validity respective to each user, p_s , where $p_s(c_i)^u$ is the subjective validity of cluster c_i for user u , and represents the number of relevant images defined by

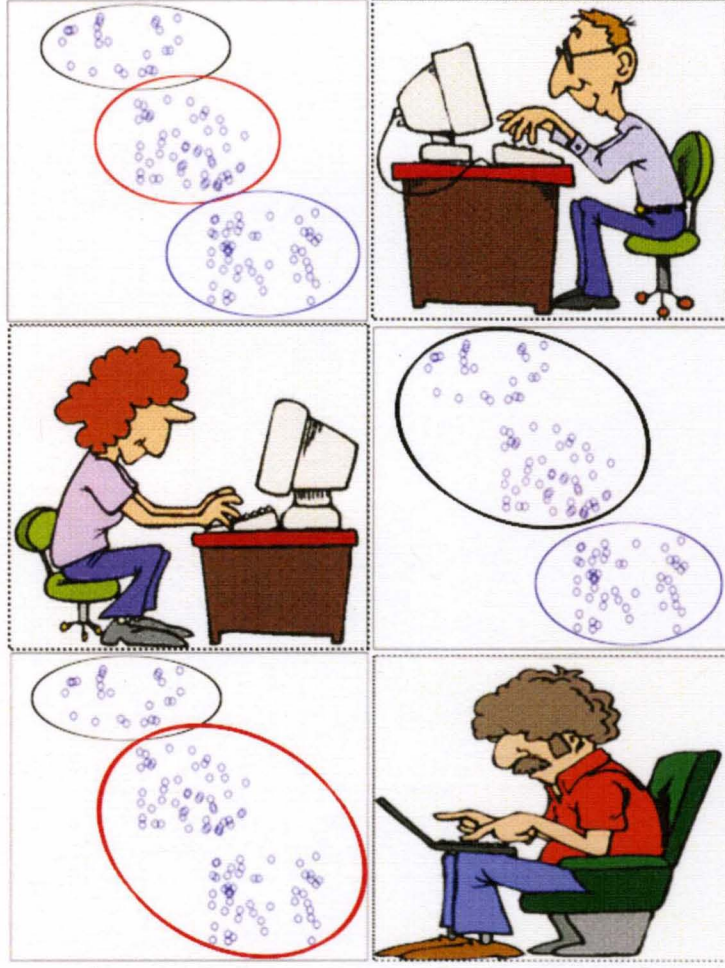


Figure 44. An example of user subjectivity.

the user in c_i . The subjective validity of a certain algorithm is then defined as the ratio of relevant images in c_i to the total number of images in that cluster, n_i for all participating user's U :

$$P^S = \frac{1}{C \cdot U} \sum_{u=1}^U \sum_{i=1}^C \frac{p_S(c_i)^u}{n_i}. \quad (79)$$

where C is the total number of clusters. Figure 46 illustrates the results of the subjective evaluations. Out of a possible score of one, $0 \leq P^S \leq 1$, the numeric values from the evaluations are given in Table 10. A total of 32 people participated in the subjective evaluation.

In Figure 47, we show examples of results from the ACC, PCCA and SCADCA

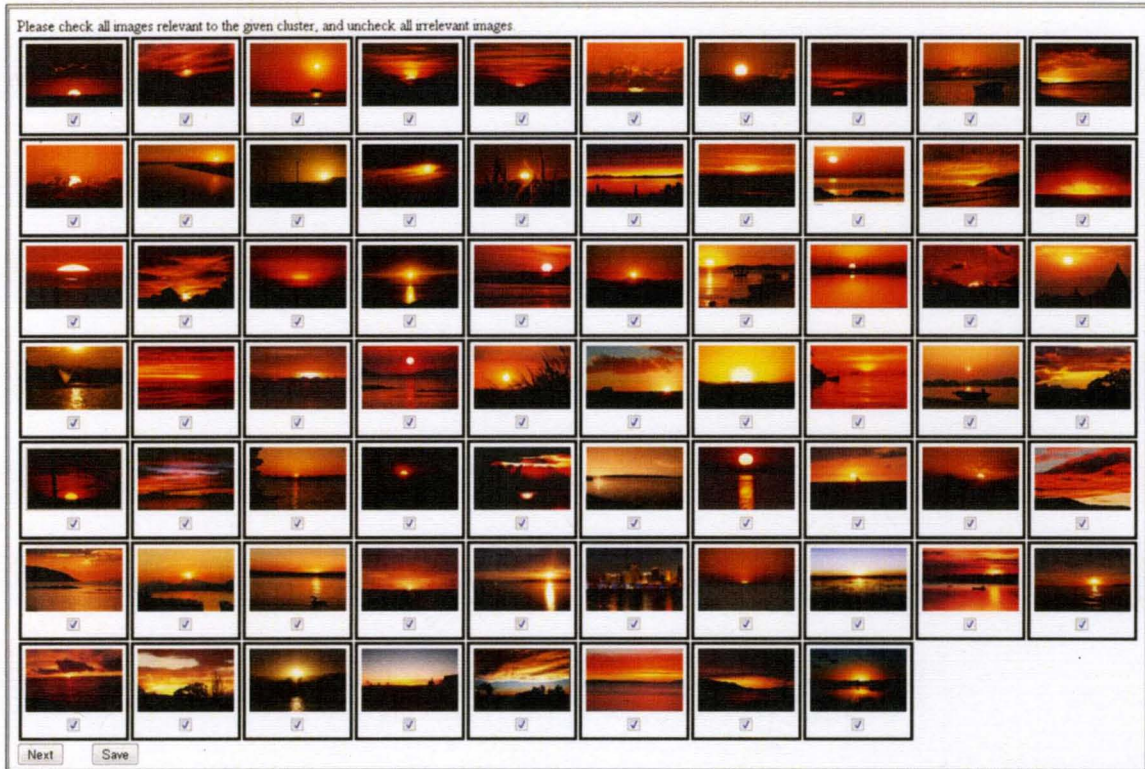


Figure 45. A screen shot of the subjective evaluation test.

TABLE 10

Validity scores for subjective evaluations.

Algorithm	P^S
ACC	0.91
PCCA	0.77
SCADCA	0.74
Random	0.35

algorithms as they would be presented to users for subjective evaluation. In this example, the ACC algorithm was able to use the partial supervision information to increase the purity of the results, and Figure 48 also illustrates an increase in performance of the ACC algorithm by utilizing the constraints. The issue of subjectivity is accurately displayed in Figure 49 and Figure 50. One user might find the inclusion of sunsets in Figure 49(b) with varying colors relevant, while others might show fa-

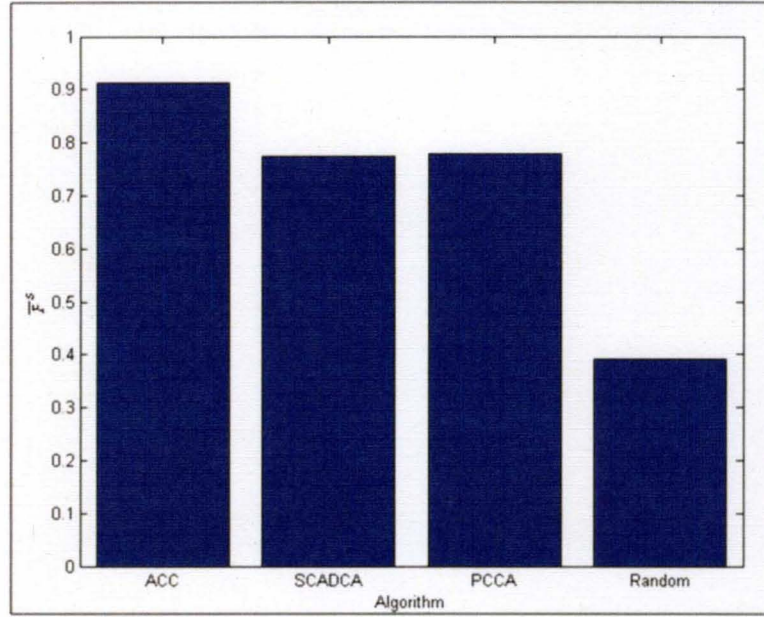


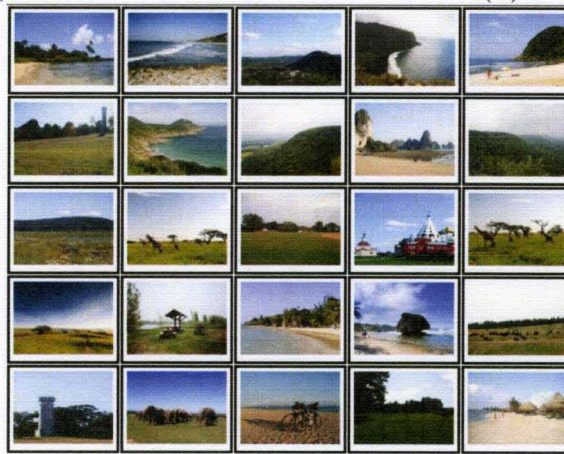
Figure 46. Validity results from subjective evaluation.

voritism towards the more uniform color spectrum in Figure 49(a), furthermore one user might also validate the results of 49(c) because all images appear to have been taken in the evening. Similar results are shown in Figure 50, where at a quick glance only shows a marginal increase of the ACC partition over the PCCA. In Figure 51, the PCCA algorithm does not provide a viable cluster for comparison, demonstrating the benefit of the relevance feature weighting in the ACC and SCADCA.



(a) ACC

(b) PCCA



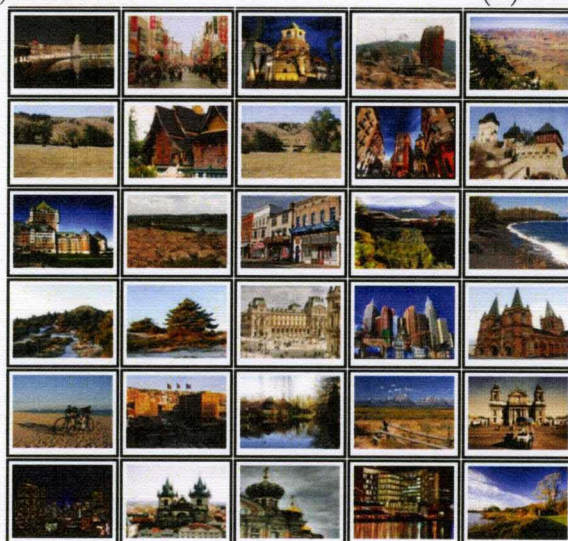
(c) SCADCA

Figure 47. Clusters utilized for subjective evaluation, where the partial supervision information provides an increase in cluster purity.



(a) ACC

(b) PCCA



(c) SCADCA

Figure 48. Using constraints, the PCCA and ACC algorithms are able to increase purity compared the SCADCA.



(a) ACC

(b) PCCA



(c) SCADCA

Figure 49. Sample result set, demonstrating the issue of subjectivity.



Figure 50. Illustrative example of subjective clusters showing forest scenes

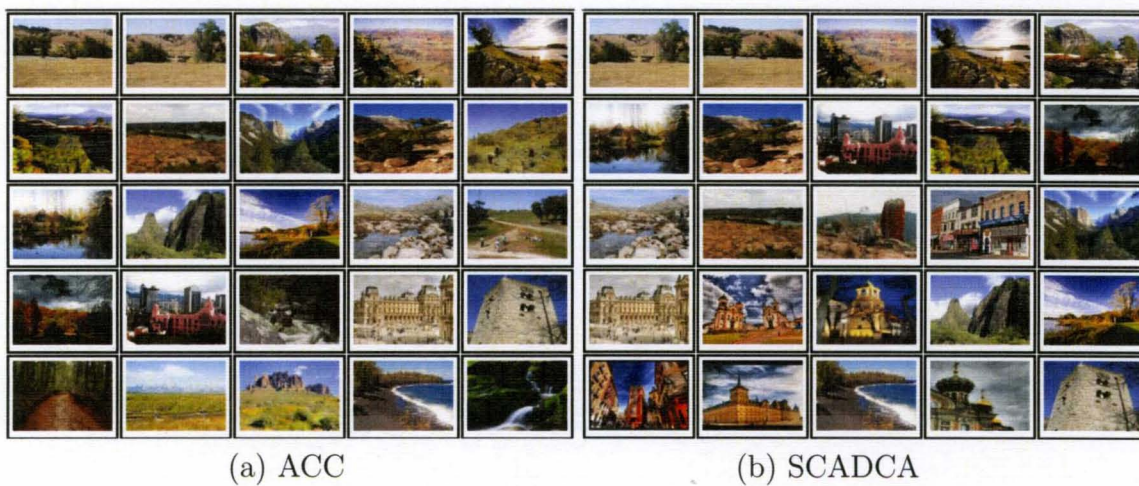


Figure 51. Subjective clusters where a strong theme may not be present.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

In this dissertation, we presented a novel clustering algorithm designed to overcome some of the conventional drawbacks suffered by partitional clustering algorithms. Our approach, called Adaptive Constrained Clustering, is a robust, dynamic, and semi-supervised algorithm. It is based on minimizing a single objective function incorporating the abilities to: (i) use multiple feature subsets while learning cluster independent feature relevance weights; (ii) search for the optimal number of clusters; and (iii) incorporate partial supervision in the form of pairwise constraints. The ACC's robustness is partially due to its generality. The algorithm allows for the use of different distance measures across any number of feature subsets which allows discovery of clusters with various shapes. To find the optimal number of clusters the ACC uses a process of competitive agglomeration. This approach does not rely on a validity measure in the search for a optimal number of clusters. It starts with a large number of small clusters and converges to the optimal number. Initializing the clustering process with a large number of clusters reduces the sensitivity of the ACC to the effects of initialization and local minima.

The partial supervision information, typically garnished from contextual information, provides the ACC with multiple levels of functionality during the clustering process. First, it assists in the discovery of a more semantically meaningful partition. By using contextual information in the creation of constraints, their inclusion provides a higher level of learning. The created clusters may have slightly differing

visual features, but a higher level of semantic similarity. Furthermore, the supervision information aids in recovering from over-agglomeration. Thus, by using a combination of unsatisfied constraints and competitive agglomeration, the ACC dynamically expands and shrinks the number of clusters at each iteration.

The proposed ACC algorithm was applied to the problem of automatically categorizing and summarizing an image database using low level visual features and high level semantic information. We showed that the ACC algorithm can incorporate partial supervision from various forms of side information, particularly from spatial, temporal, and textual metadata. Our objective and subjective experiments showed that using a soft constraint satisfaction methodology the ACC is able to partition a given data set into meaningful clusters. We also showed that our approach outperforms existing methods.

The proposed ACC algorithm uses multiple features from machine learning to perform the task of data partitioning. As the functionality of the algorithm expands so does the computational complexity. The ACC time complexity depends on the number of clusters C , the number of subsets S , the number of data samples N , and the total number of constraints M . In theory, as any of these parameters increase so does the computational time of the algorithm. However, in practice, as the number of constraints, M , increases the ACC requires fewer iterations to converge. Similarly, starting with a large number of clusters helps in better initial convergence of the feature space, and reducing the number of iterations.

The performance of the ACC algorithm is compared against two preexisting algorithms, the SCADCA and the PCCA algorithm. First, we showed a comparison of the computational complexity and convergence properties of the algorithms using synthetic data sets. Despite the ACC algorithm’s increased computational complexity, our approach’s average runtime is half that of the PCCA algorithm due to a

reduction in the number of iterations for convergence. Although the SCADCA can complete the clustering process in a substantially lower runtime, the validity results of the ACC algorithm show a 50% accuracy increase on average. Next, we demonstrated the effects of creating and utilizing constraints in the ACC and PCCA algorithms. The adaptive approach of the ACC algorithm was shown to outperform the PCCA in terms of partition accuracy over fewer iterations. Next, these three algorithms were compared using visual features gathered from an image data set. The results of the algorithms are first validated objectively, we showed that the ACC algorithm generates a 40% more accurate partition. Finally, we used subjective testing to incorporate a human element in the evaluation process. In these results, the ACC marginally outperforms the SCADCA and the PCCA algorithms.

The main limitation of the ACC is the lack of scalability. Currently, the proposed approach requires all data to be clustered to be available in memory. This may not be possible for very large data collections. Research into the area of scalable clustering [112, 113, 114] has focused on what was defined as the Merge/Purge problem [115]. Using any prototype-based algorithm, these methods buffer subsets of the database for clustering purposes. As the algorithm converges, points closest to each cluster center are purged from the buffer and refilled using new points. Subsequent passes continue to purge samples, merging the purged results into cluster subsets.

These methods cannot be easily adapted for use within the ACC algorithm. The action of purging points from the clustering process involves a loss of information and can have a detrimental effect if the purged information includes constraints. The most overhead involved with using merge/purge in the ACC is tracking the change in clusters. One of the features in the ACC is its ability to search for the optimal number of clusters. For the algorithms using the merge/purge method, once an image is purged it remains a part of its assigned cluster until the algorithm's final

convergence. If this method was used in an algorithm without a set number of clusters, multiple passes to the database would need to be made after any change in the number of clusters to update those points that were previously purged. Thus, more research is needed to develop a scalable version of the ACC algorithm. One simple approach might involve using other scalable algorithms to reduce the data set [116, 117], then use the ACC to create constraints and obtain more accurate partition.

An algorithm's robustness can be defined as its ability to handle noise and outliers, reducing the error caused by their presence during the clustering process. Currently, the ACC algorithm uses the competitive agglomeration process to handle robustness in terms of noise in a data set. In theory, using a large number of initial clusters provides the clustering process with ample locations to assign noisy data points. However, practices using this technique may have a negative effect on the computational complexity of the algorithm as the number of clusters, C , may become too large due to a lack of knowledge regarding the distribution of noise. The authors in [118, 119] proposed a method for Noise Clustering (NC) aimed at detecting noise points and assigning them to a $C + 1^{th}$ *noise cluster*. Incorporating the functionality of a NC algorithm could provide the ACC with additional robustness without the additional complexity of a large increase in the optimal number of clusters. Therefore, additional research may be necessary to develop a more robust version of the ACC algorithm.

An active area of research that provides promising approaches to narrowing the semantic gap is known as Relevance Feedback. These approaches [21, 22, 24], allow users to interact with the system by providing feedback regarding the (ir)relevance of the current retrieval results. Research into systems that use feedback information in conjunction to the functionalities of the ACC algorithm could be advantageous to Relevance Feedback systems. For example, the image database would be categorized

initially without the use of constraints. Then, using a Query by Example (QbE) approach, where the user provides an example image representing the intent of their search, the system could then use feedback from the user to create constraints. This combined approach would be advantageous in solving issues that are detrimental to each approach independently. First, the use of relevance feedback would narrow the semantic gap, where the user's intentions can influence the categorization process. For instance, a system could use relevance feedback to learn feature relevance weights and dynamically adapt the similarity measure reflecting the user's preferences. The main drawback of most relevance feedback approaches is their inability to retain knowledge from one feedback session or from one user. In other words, these systems have no long-term learning capability, where acquired information is not stored and cannot be reused to improve system performance in subsequent sessions or by other users. Incorporating feedback information in the form of constraints between image pairs is not necessarily dependent on the use of a query image. This allows for cooperation between users where partial supervision information from different sessions could be accumulated, saved, and used to continuously refine the neighborhood of the feature space during the search process.

REFERENCES

- [1] G.E. Moore, "Cramming more components onto integrated circuits.," *Electronics*, vol. 38, no. 8, pp. 114–118, 1965.
- [2] Ludicorp, "Flickr," <http://www.flickr.com/>.
- [3] Google Inc., "Panoramio," <http://www.panoramio.com/>.
- [4] V. Mezaris, I. Kompatsiaris, and M.G. Strintzis, "Region-based image retrieval using an object ontology and relevance feedback," *EURASIP Journal on Applied Signal Processing*, vol. 2004, no. 1, pp. 886–901, 2004.
- [5] C. Faloutsos and D.W. Oard, "A survey of information retrieval and filtering methods," Tech. Rep., UM Computer Science Department; CS-TR-3514, 1998.
- [6] S. Christoudoulakis, M. Theodoridou, F. Ho, M. Papa, and A. Pathria, "Multi-media document presentation, information extraction, and document formation in minos: a model and a system," *ACM Transactions on Information Systems (TOIS)*, vol. 4, no. 4, pp. 345–383, 1986.
- [7] Y. Rui and T.S. Huang, "Image retrieval: Current techniques, promising directions, and open issues," *Journal of Visual Communications and Image Representation*, vol. 10, no. 1, pp. 39–62, 1999.
- [8] C. Faloutsos, R. Barber, and M. Flickner, "Efficient and effective querying by image content," *Journal of Intelligent Information Systems*, vol. 3, no. 3/4, pp. 231–262, 1994.
- [9] M. Flickner, H. Sawhney, and W. Niblack, "Query by image and video content: the qbic system," *IEEE Computer*, vol. 28, no. 9, pp. 23–32, 1995.
- [10] B. S. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Trans. Patt. Analysis Mach. Intell.*, vol. 18, pp. 837–842, 1996.
- [11] A. Pentland, R. Picard, and S. Sclaroff, "Photobook: content-based image query system," *Int. Journal of Computer Vision*, vol. 18, no. 3, pp. 233–254, 1996.
- [12] T.S. Huang, S. Mehrotra, and K. Ramchandran, "Multimedia analysis and retrieval system (mars) project," in *Proc. of the 33rd Annual Clinic on Library Application of Data Processing*, 1996.
- [13] S. Santini, A. Gupta, and R. Jain, "Emergent semantics through interaction in image databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 3, pp. 337–351, 2001.

- [14] S. Santini and R. Jain, "Interfaces for emergent semantics in multimedia databases," in *Proceedings of SPIE Vol. 3656 Storage and Retrieval for Image and Video Databases VII*, 1999.
- [15] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "WaveCluster: A multi-resolution clustering approach for very large spatial databases," in *Proc. of the 24th Int. Conf. on Very Large Databases*, 1998, pp. 428–439.
- [16] Y. Chen, J. Z. Wang, and R. Krovetz, "Clue: Cluster-based retrieval of images by unsupervised learning," *IEEE Trans. on Image Processing*, vol. 14, no. 8, pp. 1187–1201, 2005.
- [17] S.-F. Cheng, W. Chen, and H. Sundaram, "Semantic visual templates: Linking visual features to semantics," in *IEEE Int. Conference on Image Processing*, 1998, pp. 531–535.
- [18] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Image retrieval: Ideas, influences, and trends of the new age," in *Penn State University Technical Report CSE 06-009*, 2006.
- [19] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Trans. Patt. Analysis Mach. Intell.*, vol. 22, no. 12, 2000.
- [20] H.-J. Zhang, *Multimedia Information Retrieval and Management: Technological Fundamentals and Applications (Signals and Communication Technology)*, Springer, 2003.
- [21] H.-J. Zhang, Z. Chen, W.-Y. Liu, and M. Li, "Relevance feedback in content-based image search," in *Proceedings of the 1st International Workshop on Pattern Recognition in Information Systems: in conjunction with ICEIS 2001*, 2001.
- [22] A. Kushki, P. Androutsos, K. N. Plantaniotis, and A. N. Venetsanopoulos, "Query feedback for interactive image retrieval," *IEEE Transactions On Circuits and Systems for Video Technology*, vol. 14, no. 5, pp. 644–655, 2004.
- [23] J. Urban, J. M. Jose, and C.J. van Rijsbergen, "An adaptive approach towards content-based image retrieval," in *Proc. of the Third International Workshop on Content-Based Multimedia Indexing*, 2003, pp. 119–126.
- [24] K.-H. Yap and K. Wu, "A soft relevance framework in content-based image retrieval systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 12, pp. 1557–1568, 2005.
- [25] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, "Constrained k-means clustering with background knowledge," in *Proc. of the 18th Int. Conf. on Machine Learning*, 2001, pp. 577–584.
- [26] N. Grira, M. Crucinau, and N. Boujemaa, "Active semi-supervised clustering for image database categorization," in *Content-Based Multimedia Indexing (CBMI'05)*, 2005.

- [27] S. Basu, A. Banerjee, and R.J. Mooney, "Active semi-supervision for pairwise constrained clustering.," in *Proc. of the SIAM Int. Conf. on Data Mining (SDM-2004)*, 2004, pp. 333–344.
- [28] A. Demiriz, K. Bennett, and M. Embrechts, "Semi-supervised clustering using genetic algorithms.," *Intelligent Engineering Systems Through Artificial Neural Networks*, vol. 9, pp. 809–814, 1999.
- [29] D. Stan and I. K. Sethi, "Mapping low-level image features to semantic concepts," in *Proceedings of SPIE Conference: Storage and Retrieval for Media Databases*, 2001, pp. 172–179.
- [30] T. Westerveld, "Image retrieval: Content versus context," in *Proc. of the 6th RIAO Conf. Content-Based Multimedia Information Access*, 2000.
- [31] M. Davis, S. King, N. Good, and R. Sarvas, "From context to content: leveraging context to infer media metadata," in *Proc. of the 12th Annual ACM Int. Conf. on Multimedia*, 2004, pp. 188–195.
- [32] B. S. Manjunath, P. Salembier, and T. Sikora, *Introduction to MPEG 7: Multimedia Content Description Language*, John Wiley, 2002.
- [33] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases," in *Proc. of ACM SIGMOD Int. Conf. on Management of Data*, 1998, pp. 73–84.
- [34] I. Davidson and S.S. Ravi, "Agglomerative hierarchical clustering with constraints," in *Proc. of the 9th European Conf. on Principles and Practice of Knowledge Discovery in Databases*, 2005, pp. 59–70.
- [35] Y. Zho and G. Karypis, "Hierarchical clusterin algorithms for document datasets," *Data Mining and Knowledge Discovery*, vol. 10, no. 2, pp. 141–168, 2005.
- [36] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases.," in *Proc. of the AMC SIGKDD Int. Conf. on Knowledge discovery and data mining*, 1996, pp. 226–231.
- [37] A. Hinneburg and D. Keim, "An efficient approach to clustering in large multimedia databases with noise.," in *Proc. of the 4th Int. Conf. on Knowledge Discovery and Data Mining (KDD'98)*, 1998, pp. 58–65.
- [38] O.R. Zaine and C-H. Lee, "Clustering spatial data in the presence of obstacles.," in *Proc. of the 6th Int. Symposium on Database Engineering and Applications*, 2002, pp. 214–223.
- [39] X. Wang, C. Rostoker, and H.J. Hamilton, "Density-based spatial clustering in the presence of obstacles and facilitators.," in *Proc. of the 8th European Conf. on Principles and Practice of Knowledge Discovery in Databases*, 2004, pp. 446–458.
- [40] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers and Geosciences*, vol. 10, no. 2–3, pp. 191–203, 1984.

- [41] J. MacQueen, "Some methods for classification and analysis of multivariate observations.," in *Proc. of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.
- [42] H. Frigui and R. Krishnapuram, "Clustering by competitive agglomeration," *Pattern Recognition*, vol. 30, no. 7, pp. 1223–1232, 1997.
- [43] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.
- [44] E.E. Gustafson and W.C. Kessel, "Fuzzy clustering with a fuzzy covariance matrix.," in *Proc. of the IEEE Conf. on Decision Control*, 1979, pp. 761–766.
- [45] A. Hinneburg and D. Keim, "Optimal grid-clustering: Towards breaking the curse of dimensionality," in *Proceedings of the Very Large Data Base Conference*, 1999.
- [46] C. Aggarwal and P. Yu, "Outlier detection for high dimensional data," in *Proceedings of the ACM SIGMOD Conference*, 2001.
- [47] A.K. Jain and R.C. Dubes, *Algorithms for clustering data*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.
- [48] I. Gath and G. Geva, "Unsupervised optimal fuzzy clustering.," *Pattern Anal. Machine Intel., IEEE Transaction on*, vol. 11, pp. 773–781, 1989.
- [49] S. Ray and R. H. Turi, "Determination of number of clusters in k-means clustering and application in color image segmentation.," in *Proc. of the 4th Int. Conf. on Advances in Pattern Recognition and Digital Techniques (ICAPRDT'99)*, 1999, pp. 137–143.
- [50] R.N. Dave and T. Fu, "Robust shape detection using fuzzy clustering: Practical applications," *Fuzzy Sets and Systems*, vol. 63, no. 2–3, pp. 161–185, 1994.
- [51] R. Krishnapuram, "Generation of membership functions via possibilistic clustering.," in *Proc. of the 3rd Conf. Fuzzy Systems*, 1994, pp. 902–908.
- [52] R. Krishnapuram, O. Nasraoui, and H. Frigui, "Fuzzy c spherical shells algorithm: A new approach," *Neural Networks, IEEE Transactions on*, vol. 3, pp. 663–671, 1992.
- [53] R. Krishnapuram and C. P. Freg, "Fitting an unknown number of lines and planes image data through compatible cluster merging," *Pattern Recognition*, vol. 25, no. 4, pp. 385–400, 1992.
- [54] R. N. Davés and K. J. Patel, "Progressive fuzzy clustering algorithms characteristic shape recognition," in *North Amer. Fuzzy Information Processing Soc.*, 1990, pp. 121–124.
- [55] R. Krishnapuram, H. Frigui, and O. Nasraoui, "Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation," *IEEE Trans. Fuzzy Systems*, vol. 3, no. 1, pp. 29–60, 1995.

- [56] R.N. Davé, "New measures for evaluating fuzzy partitions induced thorough c-shells clustering," in *Proc. of the SPIE Conf. on Intell. Robot Computer Vision X: SPIE*, 1991, vol. 1607, pp. 406–414.
- [57] H. Vafaie and K.D. Jong, "Robust feature selection algorithms.," in *Proc. of the 5th Int. Conf. on Tools with Artificial Intelligence(TAI'93)*, 1993, pp. 356–363.
- [58] Jihoon Yang and Vasant Honavar, "Feature subset selection using a genetic algorithm," *IEEE Intelligent Systems*, vol. 13, pp. 44–49, 1998.
- [59] Francesco Marcelloni, "Feature selection based on a modified fuzzy c-means algorithm with supervision.," *Information Sciences*, vol. 151, pp. 201–226, 2002.
- [60] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution.," in *Proc. of the 12th Int. Conf. on Machine Learning*, 2003, vol. 2, pp. 856–863.
- [61] P. Mitra, "Unsupervised feature selection using feature similarity," *Pattern Analysis and Machine Learning, IEEE Transactions on*, vol. 24, no. 3, pp. 301–313, 2002.
- [62] G. John, R. Kohavi, and K. Pfleger, "Irrelevant features and subset selection problem.," in *11th Int. Machine Learning Conference*, 1994, pp. 121–129.
- [63] Z. Su, S. Li, and H.-J. Zhang, "Extraction of feature subspaces for content-based retrieval using relevance feedback.," in *ACM Multimedia*, 2001, pp. 98–106.
- [64] C. Ding, X. He, H. Zha, and H.D. Simon, "Adaptive dimension reduction for clustering high dimensional data.," in *Proc. of the 2nd Int. Conf. on Data Mining (ICDM'02)*, 2002, pp. 147–155.
- [65] X.Z. Fern and C.E. Brodley, "Random projection for high dimensional data clustering: A cluster ensemble approach.," in *Proc. of the Int. Conf. on Machine Learning*, 2003.
- [66] K.Y. Yip, D.W. Cheung, and M.K. Ng, "On discovery of extremely low-dimensional clustering using semi-supervised projected clustering," in *Proc. of the 21st Int. Conf. on Data Engineering (ICDE'05)*, 2005, pp. 329–340.
- [67] H. Almuallim and T.G. Dietterich, "Learning with many irrelevant features.," in *Proc. of the 9th Int. Conf. on Artificial Intelligence(AAAI'91)*, 1991, vol. 2, pp. 547–552.
- [68] R. Kohavi and D. Sommerfield, "Feature subset selection using the wrapper model: Overfitting and dynamic search space topology.," in *Proc. of the 5th Int. Conf. on Knowledge Discovery and Data Mining*, 1995, pp. 1192–197.
- [69] J.G. Dy and C.E. Brodley, "Feature selection for unsupervised learning," *The Journal of Machine Learning Research*, vol. 5, pp. 845–889, 2004.
- [70] E. Bingham and H. Mannila, "Random projection in dimensionality reduction," in *Proc. of the 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'01)*, 2001, pp. 245–250.

- [71] K. Kira and L. A. Rendell, "The feature selection problem: Traditional methods and a new algorithm," in *Tenth National Conference on Artificial Intelligence*, 1992, pp. 129–134.
- [72] L. A. Rendell and K. Kira, "A practical approach to feature selection," in *International Conference on Machine Learning*, 1992, pp. 249–256.
- [73] M. Dash and H. Liu, "Feature selection for classification.," *International Journal of Intelligent Data Analysis*, vol. 1, pp. 131–156, 1997.
- [74] H. Frigui and O. Nasraoui, "Simultaneous clustering and attribute discrimination," in *Proceedings of the IEEE International Conference on Fuzzy Systems*, 2000, pp. 158–163.
- [75] H. Frigui and O. Nasraoui, "Unsupervised learning of prototypes and attribute weights," *Pattern Recognition Journal*, vol. 37, no. 3, pp. 567–581, 2004.
- [76] J. Caudill, "Bridging the semantic gap in content-based image retrieval," Tech. Rep., Ph.D proposal, University of Louisville, 2006.
- [77] H. Frigui and S. Salem, "Fuzzy clustering and subset feature weighting," in *IEEE International Conference on Fuzzy Systems*, 2003.
- [78] S. Basu, A. Banerjee, and R.J. Mooney, "Semi-supervised clustering by seeding.," in *Proc. of the 19th Int. Conf. on Machine Learning (ICML-2002)*, 2002, pp. 19–26.
- [79] N. Grira, M. Crucinau, and N. Boujemaa, "Active semi-supervised fuzzy clustering for image database categorization," in *Proc. of the 7th ACM SIGMM Int. Workshop on Multimedia Information Retrieval*, 2005, pp. 9–16.
- [80] W. Pedrycz, "Algorithms of fuzzy clustering with partial supervision," *Pattern Recognition Letters*, vol. 3, pp. 13–20, 1985.
- [81] A.M. Bensaid, L.O. Hall, J.C. Bezdek, and L.P. Clarke, "Partially supervised clustering for image segmentation," *Pattern Recognition*, vol. 29, pp. 859–872, 1996.
- [82] K. Wagstaff and C. Cardie, "Clustering with instance-level constraints.," in *Proc. of the 17th Int. Conf. on Machine Learning*, 2000, pp. 1103–1110.
- [83] D. Klein, S.D. Kamvar, and C.D. Manning, "From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering.," in *Proc. of the 19th Int. Conf. on Machine Learning (ICML'02)*, 2002, pp. 307–314.
- [84] Y.S. Abu-Mostafa, "Machines that learn from hints," *Scientific American Inc.*, 1995.
- [85] A. Dong and B. Bhanu, "Concepts learning with fuzzy clustering and relevance feedback," in *Proceedings Workshop on Machine Learning and Data Mining in Pattern Recognition*, 2001.

- [86] B. Bhanu and A. Dong, "A new semi-supervised em algorithm for image retrieval," in *CVPR*, 2003.
- [87] M. Bilenko and R.J. Mooney, "Adaptive duplicate detection using learnable string similarity measures.," in *Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining (KDD'03)*, 2003, pp. 39–48.
- [88] D. Cohn, R. Caruana, and A. McCallum, "Semi-supervised clustering with user feedback," 2003.
- [89] E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell, "Distance metric learning with application to clustering with side-information.," *Advances in Neural Information Processing Systems*, vol. 15, pp. 505–512, 2003.
- [90] R. Ge, M. Ester, W. Jin, and I. Davidson, "Constraint-driven clustering.," in *Proc. of the 13th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining (KDD'07)*, 2007, pp. 320–329.
- [91] I. Davidson and S. Ravi, "Clustering under constraints: Feasibility issues and the k -means algorithm.," in *Proc. of the 5th SIAM Int. Conf. on Data Mining (SDM'05)*, 2005.
- [92] N. Grira, M. Crucianu, and N. Boujemaa, "Unsupervised and semi-supervised clustering: a brief survey.," *A Review of Machine Learning Techniques for Processing Multimedia Content. Report of the MUSCLE European Network of Excellence*, 2004.
- [93] F. Hoppner and F. Klawonn, "A contribution to convergence theory of fuzzy c-means and derivatives," *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 5, pp. 682–694, 2003.
- [94] L. Groll and J. Jakel, "A new convergence proof of fuzzy c-means," *IEEE Trans. on Fuzzy Systems*, vol. 13, no. 5, pp. 717–721, 2005.
- [95] J.C. Bezdek, "A convergence theorem for the fuzzy isodata clustering algorithm," *IEEE Transactions on Pattern Anal. Machine Intell*, vol. 2, pp. 1–8, 1980.
- [96] R. J. Hathaway and J. Bezdek, "Recent convergence results for the fuzzy c-means clustering algorithm," *Journal of Classification*, vol. 5, no. 2, pp. 237–247, 1988.
- [97] C. Borgelt and R. Kruse, "Finding the number of fuzzy clusters by resampling," in *Proc. of the IEEE Conf. on Fuzzy Systems*, 2006, pp. 250–256.
- [98] C. Chen, G. Gagaudakis, and P. Rosin, "Similarity-based image browsing," in *IEEE International Conference on Information Visualisation*, 2000.
- [99] A. Cardoso-Cachopo and A.L. Oliveira, "Semi-supervised single-label text categorization using centroid-based classifiers," in *Proc. of the 2007 ACM Symposium on Applied Computing*, 2007, pp. 844–851.

- [100] J. Hu, M. Singh, and A. Mojsilovic, "Categorization using semi-supervised clustering," in *Proc. of the 19th Int. Conf. on Pattern Recognition (ICPR)*, 2008, pp. 1–4.
- [101] V. Estivill-Castro and I. Lee, "Fast spatial clustering with different metrics and in the presence of obstacles," in *Proc. of the 9th ACM Int Symposium on Advances in geographic information systems*, 2001, pp. 142–147.
- [102] A.K.H. Tung, J. Hou, and J. Han, "Spatial clustering in the presence of obstacles," in *Proc. of the 17th Int. Conf. on Data Engineering*, 2001, pp. 359–367.
- [103] X. Zhang, M. Wu, J. Wang, and Y. Cheng, "A novel spatial clustering with obstacles constraints based on particle swarm optimization and k-medoids," in *Proc. of the 11th Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, 2007, pp. 1105–1113.
- [104] P. Bradley, K.P. Bennett, and A. Demiriz, "Constrained k-means clustering," Tech. Rep. MSR-TR-2000-65, Microsoft Research, 2000.
- [105] A. Banerjee and J. Ghosh, "Scalable clustering algorithms with balancing constraints," in *Data Mining Knowledge Discovery*, 2006, vol. 13.
- [106] K. Toyama, R. Logan, and A. Roseway, "Geographic location tags on digital images," in *Proc. of the 11th ACM Int. Conf. on Multimedia*, 2003, pp. 156–166.
- [107] M. Naaman, Y.J. Song, A. Paepcke, and H. Garcia-Molina, "Automatic organization for digital photographs with geographic coordinates," in *Proc. of the 2004 Joint ACM/IEEE Conference on Digital Libraries*, 2004, pp. 53–62.
- [108] A. Graham, H. Garcia-Molina, A. Paepcke, and T. Winograd, "Time as essence for photo browsing through personal digital libraries," in *Proc. of the Joint Conf. on Digital Libraries*, 2002, pp. 326–335.
- [109] M. Cooper, J. Girgensohn, and L. Wilcox, "Temporal event clustering for digital photo collections," in *Proc. of the 11th ACM Int. Conf. on Multimedia*, 2003, pp. 364–373.
- [110] D.M. Squire, W. Muller, H. Muller, and T. Pun, "Content-based query of image databases: inspirations from text retrieval," *Pattern Recognition Letters*, vol. 21, no. 13-14, pp. 1193–1198, 2000.
- [111] K. Sprck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.
- [112] F. Farnstrom, J. Lewis, and C. Elkan, "Scalability for clustering algorithms revisited," *ACM SIGKDD Explorations Newsletter*, vol. 2, no. 1, pp. 51–57, 2000.
- [113] T.H. Haveliwala, A. Gionis, and P. Indyk, "Scalable techniques for clustering the web (extended abstract)," in *WebDB2000, Third International Workshop on the Web and Databases, In conjunction with ACM SIGMOD2000*, 2000.

- [114] B.L. Milenova and M.M. Campos, "O-cluster: scalable clustering of large high dimensional data sets," in *In Data Mining, Proceedings from the IEEE International Conference on*, 2002, pp. 290–297.
- [115] M.A. Hernandez and S.J. Stolfo, "The merge/purge problem for large databases," *ACM SIGMOD*, vol. 24, pp. 127–138, 1995.
- [116] P.S. Bradley and U.M. Fayyad C.A. Reina, "Scaling clustering algorithms to large databases," in *Proc. of the 4th Int. Conf. on Knowledge Discovery and Data Mining (KDD98)*, 1998, pp. 9–15.
- [117] T. Chiu, D. Fang, J. Chen, Y. Wang, and C. Jeris, "A robust and scalable clustering algorithm for mixed type attributes in large database environment," in *Proc. of the 7th Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, 2001, pp. 263–268.
- [118] R.N. Dave, "Characterization and detection of noise in clustering.," *Pattern Recognition Letters*, vol. 12, pp. 657–664, 1991.
- [119] F. Rehm, F. Klawonn, and R. Kruse, "A novel approach to noise clustering for outlier detection," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 11, no. 5, pp. 489–494, 2006.

APPENDIX A

Creation of Synthetic Data Sets

The contents of this appendix present the Matlab function for creating the synthetic datasets.

```
function [Clusters Truth] =  
    CreateSyntheticDataSet(ClusterSizes, Centers, Sig)  
%Create random Guassian distribution for each cluster  
syn = cell(length(ClusterSizes),1);  
for i = 1:length(syn)  
    syni.xy = randn(ClusterSizes(i),2);  
end  
%Modulate each cluster using defined centers and  
% sigma values  
Clusters = []; for i = 1:length(syn)  
    temp = ones(size(syn{i}.xy))*diag(Centers(i,:))';  
    syni.xy = syni.xy*Sig + temp;  
    Clusters = [Clusters;syn{i}.xy];  
end  
%Normalized each point between 0 and 1  
minXY = min(Clusters); maxXY = max(Clusters);  
for i=1:size(Clusters,1)  
    Clusters(i,:) = (Clusters(i,.)-minXY)./(maxXY-minXY);  
end  
%Create truth values for generated clusters  
Truth = []; for i = 1:length(syn)  
    Truth = [Truth,ones(1,ClusterSizes(i))*i];  
end
```

Parameters for Data Set 1

```
%Data Set 1
%Initialize cluster sizes
ClusterSizes = [100 100];
%Initialize cluster centers
Centers = [[49 51]; ...
           [55 51]];
%Initialize cluster distributions
Sig = diag([4,0.5]); ...
      diag([0.5,4]);
```

Parameters for Data Set 2

```
%Data Set 2
%Initialize cluster sizes
ClusterSizes = [100 25 75];
%Initialize cluster centers
Centers = [[49 51]; ...
           [47 53]; ...
           [51 40]];
%Initialize cluster distributions
Sig = diag([4,1]); ...
      diag([1,1]); ...
      diag([1,4]);
```

CURRICULUM VITAE

NAME: Jason D. Meredith

ADDRESS: Dept. of Computer Engineering and Computer Science
University of Louisville
Louisville, KY 40292

DOB: Jeffersonville, Indiana - April 5, 1982

EDUCATION: M.Eng., Computer Engineering and Computer Science
with Honors, May 2005
University of Louisville, Louisville, Kentucky

B.S., Computer Engineering and Computer Science
with Honors, December 2004
University of Louisville, Louisville, Kentucky

MEMBERSHIP: • ACM since 2006
• Golden Key International Honour Society since 2006
• IEEE since 2005

PUBLICATIONS: 1. H. Frigui and **J. Meredith**, "Image Database
Categorization Under Spatial Constraints
Using Adaptive Constrained Clustering,"
Int. Conf. on Fuzzy Systems (FUZZ2008), 2008

2. H. Frigui and R. Mahdi and **J. Meredith**,
"Combining Feedback and image database
categorization in CBIR,"
Int. Conf. on Multimedia and Expo (ICME2008), 2008