Object Clustering using Graphs

Bachelor Thesis

Department of Electrical and Electronics Engineering
Izmir Institute of Technology

submitted by

Kerem Aslan

from Izmir, Turkey

Supervision:

Dr. Ing. Aylin Taştan

Abstract

This work addresses the challenge of automated waste sorting using unsupervised image clustering techniques based on sparse graph representations. The primary objective is to develop a system capable of accurately categorizing waste materials without labels, overcoming the difficulties posed by high-dimensional visual data and real-world environmental noise. To this end, a comprehensive modular pipeline systematically comparing self-expressive subspace learning methods, specifically Sparse Subspace Clustering (SSC) and Block Diagonal Representation (BDR), across feature extraction techniques including Raw Pixels, Bag-of-Visual-Words (BoVW), Vector of Locally Aggregated Descriptors (VLAD), and Fisher Vectors (FV), is proposed. Experimental evaluations determine the performance of these method combinations under realistic conditions, including domain shifts with real-world images and synthetic noise injection (Gaussian, Speckle, Occlusion). The results demonstrate the critical role of feature representation, with VLAD emerging as the most robust descriptor, achieving near-perfect accuracy (0.98) under synthetic noise and maintaining superior stability (0.58 accuracy at 50% real-world replacement) by effectively preserving local structural information. Among clustering algorithms, SSC with Euclidean similarity utilizing a Gaussian RBF kernel proves to be the optimal strategy for general noise resilience, while BDR-Z specifically excels in occlusion handling with a 37.5% success rate. The study concludes that integrating VLAD features with sparse graph-based optimization provides a promising, computationally efficient approach for scalable, autonomous waste management, offering significant robustness against the unpredictable conditions of industrial deployment.

Contents

Abbreviations

Abbreviation	Meaning
ADMM	Alternating Direction Method of Multipliers
BDR	Block Diagonal Representation
BoVW	Bag of Visual Words
CC	Contrastive Clustering
CNN	Convolutional Neural Network
DEC	Deep Embedded Clustering
DINO	Self-Distillation with No Labels
EM	Expectation-Maximization
FV	Fisher Vector
GCC	Graph Contrastive Clustering
GMM	Gaussian Mixture Model
GNN	Graph Neural Network
HOG	Histogram of Oriented Gradients
ISO	International Organization for Standardization
KL	Kullback-Leibler
LRR	Low-Rank Representation
Ncut	Normalized Cuts
NJW	Ng-Jordan-Weiss
PCA	Principal Component Analysis
RBF	Radial Basis Function
ROI	Region of Interest
SCAN	Semantic Clustering by Adopting Nearest neighbors
SDG	Sustainable Development Goal
SIFT	Scale-Invariant Feature Transform
SSC	Sparse Subspace Clustering
ViT	Vision Transformer
VLAD	Vector of Locally Aggregated Descriptors

List of Figures

List of Tables

Summary of Notations

For clarity, the main mathematical notations used throughout the thesis are summarized in this section.

General Conventions: Lowercase letters denote scalars (a, b, x); lowercase bold letters denote vectors ($\mathbf{x}, \mathbf{y}$); uppercase bold letters denote matrices ($\mathbf{X}, \mathbf{A}$); calligraphic letters denote sets ($\mathcal{S}$); blackboard bold letters denote number systems ($\mathbb{R}, \mathbb{Z}$).

Symbol	**Description**
Matrices and Vectors	
$\mathbf{X} \in \mathbb{R}^{D \times N}$	Data matrix with N samples as columns, each in $\mathbb{R}^D$
$\mathbf{x}_i \in \mathbb{R}^D$	The i-th data point (column vector)
$\mathbf{C}, \mathbf{Z} \in \mathbb{R}^{N \times N}$	Coefficient matrices in sparse subspace clustering
$\mathbf{B} \in \mathbb{R}^{N \times N}$	Block-diagonal affinity matrix in BDR [1]
$\mathbf{W} \in \mathbb{R}^{N \times N}$	Affinity/similarity matrix for spectral clustering
$\mathbf{K} \in \mathbb{R}^{N \times N}$	Kernel/Gram matrix
$\mathbf{L} \in \mathbb{R}^{N \times N}$	Graph Laplacian matrix
$\mathbf{D} \in \mathbb{R}^{N \times N}$	Degree matrix (diagonal)
$\mathbf{I} \in \mathbb{R}^{N \times N}$	Identity matrix
$\mathbf{1} \in \mathbb{R}^N$	Column vector of ones
$\mathbf{A}^T$	Transpose of matrix $\mathbf{A}$
$\mathbf{\Lambda} \in \mathbb{R}^{N \times N}$	Dual variable (Lagrange multiplier) matrix in ADMM [2]

Continued on next page...

Symbol	Description
$\mathbf{h}_i \in \mathbb{R}^K$	Histogram vector for image i (BoVW) [3]
$\mathbf{v}_k \in \mathbb{R}^d$	Residual vector for visual word k (VLAD) [4]
$\mathbf{f} \in \mathbb{R}^{2Kd}$	Fisher Vector representation [5]
$\mathbf{g}_{\boldsymbol{\mu}_k}, \mathbf{g}_{\boldsymbol{\sigma}_k}$	Fisher Vector gradients w.r.t. mean and variance
Graph Theory	
$G = (V, E)$	Weighted graph with vertices V and edges E [6]
$V = \{v_1, \ldots, v_N\}$	Set of vertices (data points/images)
$E \subseteq V \times V$	Set of edges (pairwise relationships)
w_{ij}	Edge weight (similarity) between vertices i and j
Scalars and Parameters	
N	Number of data points
D	Dimensionality of feature space
d	Dimensionality of local descriptor (e.g., 128 for SIFT)
K	Number of clusters, visual words, or GMM components
k	Index for clusters or subspaces
i, j	Indices for data points
λ, γ, ρ	Regularization and penalty parameters
σ	Bandwidth parameter for Gaussian RBF kernel
π_k	Mixture weight for k-th Gaussian component
GMM Parameters (Fisher Vector)	
$\boldsymbol{\mu}_k$	Mean vector of k-th Gaussian component
$\boldsymbol{\sigma}_k$	Standard deviation vector (diagonal covariance)
$\boldsymbol{\Sigma}$	Covariance matrix

Continued on next page...

Symbol	Description
$\gamma_k(\mathbf{x}_i)$	Posterior probability of $\mathbf{x}_i$ belonging to component k
Norms and Operators	
$\|\mathbf{x}\|_1$	ℓ_1-norm: $\sum_i \|x_i\|$
$\|\mathbf{x}\|_2$	ℓ_2-norm (Euclidean): $\sqrt{\sum_i x_i^2}$
$\|\mathbf{X}\|_F$	Frobenius norm: $\sqrt{\sum_{i,j} x_{ij}^2}$
$\|\mathbf{B}\|_{\boxed{k}}$	k-block diagonal regularizer
$\text{diag}(\mathbf{X})$	Diagonal vector of matrix $\mathbf{X}$
$\text{Diag}(x_1, \ldots, x_N)$	Diagonal matrix with entries on diagonal
$\text{sign}(x)$	Sign function returning -1, 0, or 1
$\|\mathbf{X}\|$	Element-wise absolute value
$\mathcal{S}_\tau(\cdot)$	Soft-thresholding operator with threshold τ
Functions	
$NN(\mathbf{x}_i)$	Index of nearest neighbor (visual word) for $\mathbf{x}_i$
$\lambda_i(\mathbf{M})$	i-th eigenvalue of matrix $\mathbf{M}$
$\exp(\cdot)$	Exponential function
$\text{Tr}(\mathbf{A})$	Trace of matrix $\mathbf{A}$
Sets and Spaces	
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^{D \times N}$	Space of $D \times N$ real matrices
$\mathcal{S}_k$	k-th subspace
Special Symbols	
$\hat{\mathbf{x}}$	Estimate or approximation of $\mathbf{x}$
$\leftarrow$	Assignment operator (e.g., $x \leftarrow f(x)$)

Continued on next page...

Symbol	Description
$\mathbf{1}[\cdot]$	Indicator function

Chapter 1

Introduction

1.1 Motivation

Efficient waste management stands as one of the most critical environmental challenges of the 21st century. With rapid urbanization and industrial growth, global waste generation is projected to increase significantly, reaching approximately 3.40 billion tonnes by 2050 [7]. In this context, recycling plays a crucial role in conserving natural resources, reducing the volume of waste sent to landfills, and mitigating environmental pollution.

However, the success of recycling depends heavily on the accurate separation of materials at the source to ensure high recovery rates and material quality [8]. Existing methods, which rely largely on the manual sorting of waste, present several significant drawbacks [9]. Manual sorting requires a significant amount of labor force, which is relatively slow, and is prone to human error due to the complexity and variety of waste materials [10]. Furthermore, the lack of consistent public engagement in source separation increases the inefficiency of existing systems [7]. Consequently, automating the classification of recyclable objects via artificial intelligence and machine learning has become essential for developing modern, scalable, and efficient waste management systems [9]. Recent studies demonstrate that automated systems, particularly those utilizing deep learning, convolutional neural network architectures and computer vision, can significantly outperform manual sorting in terms of both speed and accuracy [10].

To address this need, this thesis focuses on automating the sorting process by grouping object images into meaningful clusters using graph theory and clustering algorithms. The primary motivation is to eliminate the inefficiencies of manual sorting by developing a system capable of accurately assigning images of distinct waste materials, such as plastic bottles, glass containers, metal cans and cardboard boxes; to their respective clusters. By leveraging the structural properties of data

through graph-based methods, this study aims to contribute to the development of automated waste sorting technologies that can operate effectively without extensive human intervention.

This research is designed with a commitment to global sustainability standards, directly addressing the United Nations Sustainable Development Goals (SDGs) [11]. Specifically, the technical objectives and social impact of this thesis align with SDG 9 (Industry, Innovation and Infrastructure) and SDG 12 (Responsible Consumption and Production). The developed object clustering system plays an essential role in advancing industrial automation technology for recycling and waste labeling, supporting innovation for modern recycling facilities (SDG 9). Furthermore, by focusing on improving automated waste sorting to enhance the efficiency, speed, and accuracy of recycling processes, the project contributes to more sustainable consumption and production patterns (SDG 12).

1.2 State-of-the-Art

Image clustering, the unsupervised grouping of visual data based on similarity, stands as a fundamental problem in computer vision and machine learning. The primary objective is to maximize the intra-class similarity while ensuring significant separation between different clusters [12]. While traditional methods relied heavily on manual feature engineering and standard partitioning algorithms [13], the field has evolved significantly with the advent of high-dimensional data in areas such as social media, medical imaging and autonomous systems. Current research focuses on overcoming the limitations of Euclidean distance in high-dimensional spaces and eliminating the dependence on domain-specific manual feature extraction through advanced representation learning [14]. This section reviews the progression from sparse graph-based methods to the latest deep learning-based clustering paradigms.

1.2.1 Sparse Graph Representations

Graph-based clustering relies on constructing an affinity matrix that captures the pair-wise similarities between data points [15]. A critical challenge in this domain is constructing a graph that represents the underlying data manifold, especially when the data is high-dimensional and contaminated with noise [16]. Sparse graph representations address this by enforcing sparsity constraints, ensuring that each data point connects only to a small subset of its most relevant neighbors. State-of-the-art methods include Sparse Subspace Clustering (SSC) [17], which utilizes ℓ_1-norm minimization to express each data point as a linear combination of others

within the same subspace. Other approaches, such as Low-Rank Representation (LRR) [18], and Block Diagonal Representation (BDR) [1], impose global rank or block-diagonal constraints to enhance the clustering structure, making them robust to outliers and noise.

1.2.2 Graph-Based Clustering Algorithms

Once the affinity graph is constructed, the next step involves partitioning the graph into disjoint subgraphs, where each subgraph corresponds to a cluster. Spectral Clustering [19, 15] is widely regarded as the most effective algorithm for this task. It leverages the eigenvalues and eigenvectors of the graph Laplacian matrix to map data points into a lower-dimensional embedding space where they are linearly separable. To improve partition quality, objectives like Normalized Cuts (Ncut) [20] are minimized to ensure that clusters are balanced and edges between different clusters are minimized. Recent advancements also explore bipartite graph matching and tensor-based clustering to handle multi-view data and complex structural dependencies.

1.2.3 Visual Representation and Feature Extraction

The quality of any clustering pipeline is limited by the quality of the input features. Traditional approaches relied on handcrafted local descriptors such as Scale-Invariant Feature Transform (SIFT) [21] and Histogram of Oriented Gradients (HOG) [22]. These were typically aggregated into global representations using methods like Bag of Visual Words (BoVW) and Vector of Locally Aggregated Descriptors (VLAD) [4, 23], which encode the distribution of local features.

However, the field has undergone a paradigm shift with the advent of Deep Learning. Convolutional Neural Networks (CNNs), such as ResNet [24] and VGG [25], extract hierarchical semantic features directly from raw images. More recently, Foundation Models based on the Transformer architecture have set new benchmarks. Vision Transformers (ViT) [26] process images as sequences of patches, capturing long-range global dependencies. In the unsupervised domain, self-supervised learning frameworks like DINO [27], DINOv2 [28], and the most recent DINOv3 [29] have demonstrated exceptional capability. These methods utilize self-distillation and masked image modeling to learn powerful, semantic-rich representations without the need for human annotation.

1.2.4 Deep Clustering Paradigms

To overcome the limitations of separating feature extraction and clustering into disjoint stages, Deep Clustering methods have emerged, aiming to jointly perform representation learning and clustering [14]. By integrating deep neural networks with clustering objectives, these methods automate feature extraction and efficiently handle large-scale, high-dimensional data. According to the recent survey by Hou et al. [14], deep image clustering can be categorized into three main paradigms:

- **Self-Training based Clustering:** Pioneered by the Deep Embedded Clustering (DEC) algorithm [30], these methods typically employ autoencoders to learn latent representations. A clustering layer is added to the encoder output, and the network is optimized by minimizing the Kullback-Leibler (KL) divergence between the soft assignments and an auxiliary target distribution. This iterative process refines both the feature space and the cluster centers simultaneously.

- **Contrastive Learning based Clustering:** Leveraging the success of self-supervised learning, methods like Contrastive Clustering (CC) [31] and Graph Contrastive Clustering (GCC) [32] maximize the similarity between positive pairs (e.g., augmented views of the same image) while minimizing the similarity between negative pairs. Unlike standard contrastive methods, these approaches extend the contrastive objective to the cluster level, ensuring high intra-cluster compactness and inter-cluster separability [14].

- **Pseudo-Labeling based Clustering:** Algorithms such as SCAN [33] utilize high-confidence predictions from the network as pseudo-labels to guide the training process. This creates a self-reinforcing loop where improved feature representations lead to more accurate pseudo-labels, which in turn further refine the features.

Despite their impressive performance, deep learning-based clustering methods come with several notable limitations. First, they typically require substantial computational resources, including high-end GPUs and extended training times, which may be prohibitive for resource-constrained industrial applications [14]. Second, end-to-end deep clustering models often function as "black boxes," lacking the interpretability and theoretical guarantees provided by classical optimization-based methods [34]. Third, while self-supervised methods reduce the need for explicit labels, they still require large amounts of unlabeled data to learn effective representations, and their training dynamics can be unstable or sensitive to hyperparameter

choices. Finally, the joint optimization of feature learning and clustering objectives can lead to degenerate solutions, such as trivial cluster assignments or mode collapse, requiring careful architectural design and regularization strategies.

In contrast, sparse graph-based methods like SSC and BDR offer well-understood mathematical formulations with provable recovery guarantees under certain conditions [16, 17]. These methods are modular, allowing the feature extraction and clustering stages to be analyzed and optimized independently. This thesis focuses on evaluating the robustness of such sparse graph representations (Section 1.2.1) when combined with various feature extractors, providing a bridge between rigorous mathematical subspace modeling and modern representation learning while maintaining computational efficiency and interpretability.

1.3 Problem Statement

Developing an effective automated waste sorting system using unsupervised image clustering presents several challenges that this thesis aims to address:

- **High-Dimensional Feature Spaces:** Modern feature extractors, particularly deep foundation models, produce high-dimensional embedding vectors that capture rich semantic information. However, operating in such high-dimensional spaces comes with significant computational and statistical challenges, a phenomenon known as the curse of dimensionality [35]. As dimensionality increases, the distance between data points becomes increasingly uniform, degrading the discriminative power of similarity metrics. Selecting appropriate feature representations that balance descriptiveness with computational tractability is therefore a critical design decision.

- **Real-World Noise and Domain Shift:** Images captured in industrial waste sorting environments are subject to varying lighting conditions, occlusions, background clutter, and sensor noise. Furthermore, a significant domain gap often exists between controlled laboratory datasets used for development and the unpredictable conditions of real-world deployment, as illustrated in Figure 3.1. Any practical system must demonstrate robustness to such noise and domain shifts, maintaining accurate clustering performance even when input data deviates substantially from ideal conditions.

- **Computational Scalability:** Subspace clustering methods like SSC [17] and LRR [18] have demonstrated theoretical robustness to sparse noise and outliers. However, their computational complexity, often cubic in the number

of data points ($\mathcal{O}(N^3)$), limits their scalability to large datasets. Efficiently processing thousands of images while maintaining clustering accuracy is a critical requirement for industrial deployment.

- **Unsupervised Operation Requirement:** A defining constraint of the problem addressed in this thesis is the requirement for fully unsupervised operation. Obtaining large-scale, accurately labeled datasets for every waste category is prohibitively expensive and time-consuming [9]. This necessitates clustering algorithms that can discover meaningful group structures purely from the intrinsic properties of the data, without any prior supervision, making the accurate recovery of cluster assignments a non-trivial task.
- **Algorithm and Feature Selection:** Given the variety of available feature extraction methods (raw pixels, BoVW, VLAD, Fisher Vectors) and subspace clustering algorithms (SSC, BDR variants) with different similarity measures (linear kernel, Gaussian RBF kernel), identifying the optimal combination for waste image clustering is not straightforward. A systematic comparison is required to determine which configurations yield the best trade-off between clustering accuracy, noise robustness, and computational efficiency.

To address all the problems listed above, this thesis proposes a comprehensive modular pipeline that systematically evaluates sparse graph-based subspace clustering methods across multiple feature representations and similarity measures, with particular emphasis on robustness to real-world noise conditions. The details of the proposed approach are presented in the following section.

1.4 Creative Solution Development

To address the challenges of automated waste sorting identified in Section 1.3, three distinct methodological approaches were considered during the design phase of this thesis. Each approach represents a different trade-off between technical complexity, computational requirements, and expected clustering accuracy.

The first approach, serving as a baseline, relies on traditional descriptor-based feature extraction combined with standard graph clustering. In this framework, raw images are processed to extract handcrafted features such as Histograms of Oriented Gradients (HOG) [22] or color histograms. A similarity graph is then constructed using k-nearest neighbors based on Euclidean distances, followed by conventional graph partitioning algorithms. While this approach benefits from well-established implementations and low computational overhead, it is inherently sensitive to variations in lighting, background, and object orientation.

The second approach leverages Graph Neural Networks (GNNs), which have demonstrated remarkable success in learning representations directly on graph-structured data. In this framework, an initial affinity graph is constructed from the images, and a GNN architecture such as a Graph Convolutional Network iteratively refines node embeddings to capture complex non-linear relationships. However, this deep learning-based approach requires substantial training data to avoid overfitting and demands significant computational resources, presenting a high technical risk given the moderate size of the target dataset.

The third approach, which forms the core methodology of this thesis, focuses on learning the graph structure itself through sparse self-representation. Rather than manually defining the affinity graph, algorithms such as Sparse Subspace Clustering (SSC) [17] and Block Diagonal Representation (BDR) [1] are employed to mathematically derive the optimal graph topology from the data. This approach is combined with robust global descriptors such as Bag of Visual Words (BoVW) and Fisher Vectors [5] to capture discriminative visual information invariant to environmental variations.

Following a comparative evaluation of these alternatives based on technical feasibility, innovation potential, and alignment with sustainability goals, this thesis adopts a two-stage methodology. The first approach is implemented as an essential baseline for benchmarking purposes, while the third approach serves as the primary technical contribution. This selection is justified by the ability of sparse graph learning methods to achieve high clustering accuracy without the prohibitive computational and energy costs associated with deep learning alternatives, thereby aligning with the environmental sustainability objectives outlined in ISO 14001 [36].

1.5 Project Scope and Methodology

The research conducted in this thesis was organized into six interconnected work packages, each addressing a specific aspect of the development pipeline.

The first phase focused on establishing a comprehensive theoretical foundation through an extensive literature review. This investigation covered three key domains: unsupervised clustering algorithms, including traditional k-means and Spectral Clustering [19] as well as advanced Graph Learning frameworks such as SSC [17] and BDR [1]; visual feature extraction methods including SIFT [21], BoVW [3], and Fisher Vectors [5]; and similarity measures for affinity matrix construction.

The second phase involved the systematic collection of image datasets featuring recyclable objects distributed across 16 distinct categories. The primary dataset comprises 533 images captured under controlled conditions with uniform illumina-

tion and clean white-black backgrounds, establishing an ideal baseline for algorithm evaluation. Additionally, a real-world dataset of 419 images was collected featuring natural background variations, diverse lighting conditions, and varying camera distances to evaluate robustness against domain shift.

The third phase addressed the core algorithm implementation, developing the complete Graph Learning pipeline in MATLAB. This encompassed image vectorization, sparse self-representation optimization using the Alternating Direction Method of Multipliers (ADMM) [2], affinity matrix construction, and spectral clustering for final cluster assignment.

The fourth phase focused on feature engineering, transitioning from raw pixel representations to discriminative global descriptors including BoVW, VLAD [4], and Fisher Vectors. Additionally, two similarity measures were implemented for affinity matrix construction: linear kernel (inner product) and Gaussian RBF kernel based on Euclidean distance.

The fifth phase conducted comprehensive experiments comparing the baseline raw-pixel approach with the advanced feature-based pipeline. This included systematic hyperparameter optimization through grid search, quantitative evaluation using clustering accuracy and F1-score metrics, and analysis of computational performance.

The final phase consolidated the research findings into this thesis document, with particular emphasis on visualizing the learned affinity matrices to demonstrate the block-diagonal structure that underlies successful subspace clustering.

Several technical risks were identified and addressed throughout the project. The risk of overfitting to specific environmental conditions was mitigated through dataset diversification with both controlled and real-world images. The potential inadequacy of handcrafted features was addressed by prioritizing scale and rotation-invariant descriptors such as SIFT. Computational complexity concerns were managed through efficient iterative solvers such as ADMM. Finally, the challenge of hyperparameter sensitivity was tackled through structured grid search optimization.

From an economic perspective, this research was conducted as a software-based project with minimal material costs. The primary resources, including computing hardware and MATLAB licenses, were provided through university. The value of this work lies in its potential application to industrial waste management systems, where automated sorting can reduce labor costs and improve recycling efficiency.

1.6 Engineering Standards and Design Constraints

The design and implementation of the proposed clustering system were guided by established engineering standards and an analysis of realistic constraints to ensure validity, sustainability, and practical applicability.

From a quality management perspective, the principles of ISO 9001 [37] informed the systematic documentation of all experiments and code implementations. This framework ensured that performance comparisons across different feature extraction methods and clustering algorithms are valid, repeatable, and thoroughly documented, enabling independent verification of the reported results.

The environmental management principles outlined in ISO 14001 [36] played a significant role in shaping the methodological decisions of this thesis. Given that the main application domain is waste management and recycling, it was considered essential that the proposed solution itself adheres to principles of environmental sustainability. This alignment directly supports SDG 12 (Responsible Consumption and Production) by enabling more efficient recycling processes. The standard influenced the decision to favor computationally efficient algebraic methods over energy-intensive deep learning alternatives, ensuring that the clustering system does not contradict its own sustainability objectives. Furthermore, by contributing to industrial automation in recycling (SDG 9), the project supports the broader goal of upgrading technological capabilities in waste management infrastructure [11].

Several realistic constraints were also considered during the system design. From an economic standpoint, the solution was constrained to operate on standard computing hardware without requiring expensive specialized equipment such as high-end GPUs typically necessary for training large neural networks. This constraint ensures that the proposed methodology remains accessible to academic researchers and industrial practitioners with limited computational resources. Environmental constraints demanded that the selected algorithms be energy-efficient, avoiding high computational loads that would contradict the goal of sustainable waste management. From an ethical and social perspective, all referenced research is properly cited, and the project aims to benefit society by automating repetitive and unpleasant tasks in recycling facilities. Finally, scalability considerations ensured that the system architecture remains modular, allowing future researchers to substitute optimization solvers, feature extraction methods, or even the clustering algorithm itself without requiring a complete redesign of the pipeline.

1.7 Proposed Solution

Based on the creative solution development and selected approach, this thesis proposes a comprehensive and modular image clustering pipeline. As established in the Problem Statement, the primary objective is to identify the most effective combination of components for accurate, unsupervised categorization of waste material images, specifically addressing high dimensionality and real-world noise.

The proposed solution implements a Graph Learning framework using Sparse Subspace Clustering (SSC) and Block Diagonal Representation (BDR). The approach is built upon the principle of self-expressiveness [17], which states that each data point lying on a union of subspaces can be efficiently represented as a sparse linear combination of other points from the same subspace. This property is used to construct a sparse affinity graph, which is subsequently partitioned using spectral clustering [19].

The pipeline consists of five main stages:

1. **Data Loading and Preprocessing:** Images are loaded from the dataset, resized to a uniform resolution, and converted to a consistent format.

2. **Feature Extraction:** To handle environmental variations (Proposal 3's key advantage), four distinct feature extraction methods are systematically compared: Raw Pixel features, Bag-of-Visual-Words (BoVW) [3], Vector of Locally Aggregated Descriptors (VLAD) [4], and Fisher Vectors [5].

3. **Similarity Measure Selection:** Two similarity measures are evaluated for constructing the Gram matrix: the linear kernel (inner product) and the Gaussian Radial Basis Function (RBF) kernel.

4. **Subspace Clustering:** The feature matrix is passed to either Sparse Subspace Clustering (SSC) [17] or Block Diagonal Representation (BDR) [1], solved via ADMM [2] to learn the affinity matrix.

5. **Spectral Clustering and Evaluation:** The affinity matrix is partitioned using the Ng-Jordan-Weiss (NJW) spectral clustering algorithm. Performance is evaluated using clustering accuracy and F1-score.

To evaluate practical applicability, the pipeline is further tested under realistic noise conditions by augmenting the clean dataset with images captured in uncontrolled environments and applying artificial noise types.

1.8 Thesis Overview

This thesis is organized as follows to provide a clear and structured presentation of the research:

Chapter 1 (the current chapter) presents the motivation for automated waste sorting, reviews the state-of-the-art in sparse graph representations, clustering algorithms, and visual feature extraction. It also states the problem, introduces the proposed method, and provides an overview of the thesis structure.

Chapter 2 introduces the mathematical foundations underpinning the proposed methodology. It covers basic definitions of graphs and affinity matrices, similarity measures including inner product and Gaussian RBF kernels, sparse representation techniques (SSC and BDR), spectral clustering, and the feature extraction methods (Raw Pixel, BoVW, VLAD, and Fisher Vectors) used in this work.

Chapter 3 details the implementation of the proposed image clustering pipeline. It describes the dataset used for evaluation, the complete pipeline architecture, the specific parameters for each feature extraction and clustering method, the evaluation metrics (clustering accuracy and F1-score), and the noise robustness evaluation protocol.

Chapter 4 presents the experimental results and analysis. It provides a systematic comparison of SSC and BDR methods across different feature extraction approaches and similarity measures, evaluates noise robustness and discusses the computational performance of each configuration.

Chapter 5 summarizes the key findings of this thesis, discusses the limitations of the proposed approach, and outlines potential directions for future research.

The relevant project source codes for this thesis are available at: `https://github.com/Kerem-Aslan/object-clustering-graphs/`

Chapter 2

Fundamental Concepts and Theoretical Background

2.1 Basic Definitions

Graph-based clustering methods rely on representing data as a graph where vertices correspond to data points and edges encode relationships between them. This section introduces the fundamental concepts of graphs and affinity matrices that underpin the spectral clustering framework employed by both SSC and BDR methods.

2.1.1 Graphs and Affinity Matrices

A graph $G = (V, E)$ consists of a set of vertices $V = \{v_1, v_2, \ldots, v_N\}$ and a set of edges $E \subseteq V \times V$ [6]. In the context of image clustering, each vertex represents an image and edges encode pairwise relationships. A weighted graph extends this definition by associating a non-negative weight $w_{ij} \geq 0$ with each edge, where the weight typically represents similarity between vertices.

The affinity matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ provides the matrix representation of a weighted graph. Each entry w_{ij} quantifies the similarity between data points $\mathbf{x}_i$ and $\mathbf{x}_j$. For undirected graphs, this matrix is symmetric ($\mathbf{W} = \mathbf{W}^T$) and has non-negative entries. The diagonal entries are typically set to zero to avoid self-loops. The quality of the affinity matrix directly determines clustering performance, as spectral clustering algorithms operate on the graph structure encoded by $\mathbf{W}$ [15]. Figure 2.1 illustrates this graph-based representation where images are treated as nodes and edge weights encode visual similarity.

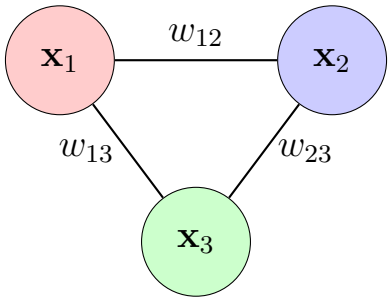


Figure 2.1: Illustration of a weighted graph $G = (V, E)$ for image clustering. Each node represents an image (data point $\mathbf{x}_i$), and edge weights w_{ij} encode the pairwise similarity between images. In subspace clustering, these weights are derived from a sparse coefficient matrix learned through self-expressiveness.

2.1.2 Affinity Matrix Construction

Since the affinity matrix $\mathbf{W}$ forms the foundation for spectral clustering, different similarity measures lead to different graph structures and clustering outcomes. This section describes the common approaches for constructing the affinity matrix based on pairwise similarities between data points.

Affinity Matrix Construction using Inner Product

The simplest measure of similarity between two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ is their inner product (linear kernel):

$$s(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j \tag{2.1}$$

The matrix of all pairwise inner products is the Gram matrix $\mathbf{G} = \mathbf{X}^T\mathbf{X}$. When features are ℓ_2-normalized, this measure is equivalent to cosine similarity.

Affinity Matrix Construction using Gaussian Radial Basis Function (RBF) Kernel

To capture non-linear relationships, the Gaussian Radial Basis Function (RBF) kernel is often used. It is defined based on the Euclidean distance between points:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right) \tag{2.2}$$

where σ is a scaling parameter that controls the width of the kernel. This kernel maps the data into an infinite-dimensional feature space, allowing linear algorithms to discover non-linear patterns [38]. This measure is particularly relevant for the Euclidean variants of the subspace clustering algorithms implemented in this work.

Affinity Matrix Construction using Cosine Similarity

Cosine similarity is a measure of similarity between two vectors based on the angle between them and normalized by the vector magnitudes. It is defined as:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\|_2 \|\mathbf{x}_j\|_2} \tag{2.3}$$

where θ is the angle between the vectors. This metric is utilized in the construction of k-nearest neighbor graphs and in the K-means clustering of Bag-of-Visual-Words features.

2.2 Sparse Affinity Matrix Construction

While the similarity measures described above can construct affinity matrices directly from pairwise distances, these dense matrices may not effectively reveal the underlying subspace structure of high-dimensional data. Sparse affinity matrices offer several advantages for subspace clustering [17, 1]:

- **Block-diagonal structure:** When data points lie in a union of subspaces, an ideal affinity matrix should exhibit block-diagonal structure, where non-zero entries only appear between points belonging to the same subspace. Sparse representations naturally encourage this structure.
- **Noise robustness:** Sparsity constraints act as regularizers that reduce the influence of noisy or corrupted samples, as the optimization tends to select only the most relevant neighbors for reconstruction.
- **Computational efficiency:** Sparse matrices require less storage and enable faster spectral decomposition compared to dense alternatives.
- **Theoretical guarantees:** Under certain geometric conditions (such as sufficient incoherence between subspaces), sparse representation methods provide provable recovery guarantees [16].

The key idea behind sparse affinity matrix construction is the *self-expressiveness property*: each data point lying on a subspace can be represented as a linear combination of other points from the same subspace [17]. Mathematically, given a data matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$, each data point $\mathbf{x}_i$ can be expressed as:

$$\mathbf{x}_i = \mathbf{X}\mathbf{z}_i \tag{2.4}$$

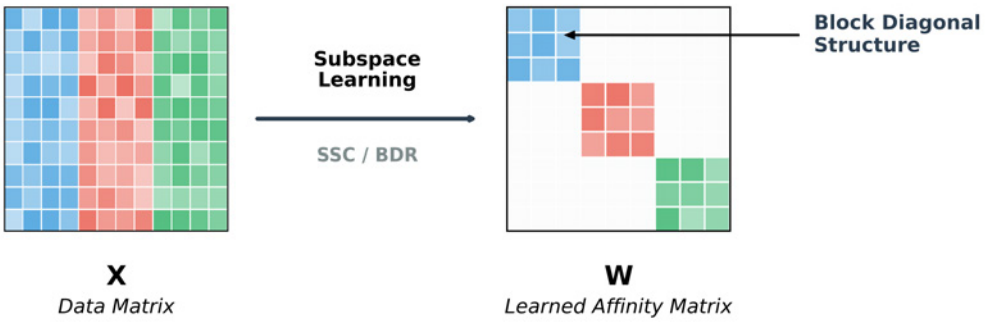


Figure 2.2: Illustration of affinity matrix construction using subspace learning methods (SSC, BDR). The data matrix $\mathbf{X}$ contains samples from k different subspaces (represented by different colors). Through self-expressiveness optimization, the learned affinity matrix $\mathbf{W}$ exhibits an ideal block-diagonal structure where non-zero entries connect only samples from the same subspace.

where $\mathbf{z}_i \in \mathbb{R}^N$ is a coefficient vector with $z_{ii} = 0$ (to avoid trivial self-representation). The full representation can be written compactly as $\mathbf{X} = \mathbf{XZ}$, where $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N]$ is the coefficient matrix with zero diagonal.

The key challenge is computing the coefficient matrix $\mathbf{Z}$ such that it reveals the underlying subspace structure. Ideally, $\mathbf{Z}$ should be block-diagonal, where each block corresponds to data points from the same subspace [1]. Once $\mathbf{Z}$ is computed, the sparse affinity matrix is constructed through symmetrization:

$$\mathbf{W} = |\mathbf{Z}| + |\mathbf{Z}|^T \tag{2.5}$$

Figure 2.2 illustrates this process, showing how a data matrix $\mathbf{X}$ is transformed into a block-diagonal affinity matrix $\mathbf{W}$ through the self-expressiveness principle. The resulting block-diagonal structure ensures that points belonging to the same subspace are strongly connected while cross-subspace connections are minimized.

In the following subsections, two popular sparse affinity matrix construction methods are explained: Sparse Subspace Clustering (SSC) and Block Diagonal Representation (BDR).

2.2.1 Sparse Subspace Clustering (SSC)

Sparse Subspace Clustering (SSC) formulates subspace clustering as a sparse representation problem. The core idea is that each data point should be represented by a sparse combination of other data points. Under certain geometric conditions on the subspaces (such as sufficient incoherence between subspaces), this sparse representation will select points from the same subspace [17, 16].

The SSC optimization problem is formulated as:

$$\min_{\mathbf{C}} \|\mathbf{C}\|_1 \quad \text{s.t.} \quad \mathbf{X} = \mathbf{X}\mathbf{C},\ \text{diag}(\mathbf{C}) = \mathbf{0} \tag{2.6}$$

where $\mathbf{X}$ is the data matrix, and $\|\mathbf{C}\|_1 = \sum_{i,j} |c_{ij}|$ is the ℓ_1-norm that promotes sparsity in the coefficient matrix $\mathbf{C}$. The constraint $\text{diag}(\mathbf{C}) = \mathbf{0}$ eliminates the trivial solution of representing a point as a linear combination of itself, enforcing the self-expressiveness property [17].

For real-world data corrupted by noise and sparse outliers, the formulation is extended. According to Elhamifar and Vidal [17], data can be modeled as $\mathbf{X} = \mathbf{X}\mathbf{C} + \mathbf{E} + \mathbf{Z}$, where $\mathbf{E}$ represents sparse outlying entries and $\mathbf{Z}$ represents noise:

$$\min_{\mathbf{C},\mathbf{E},\mathbf{Z}} \|\mathbf{C}\|_1 + \lambda_e \|\mathbf{E}\|_1 + \frac{\lambda_z}{2}\|\mathbf{Z}\|_F^2 \quad \text{s.t.} \quad \mathbf{X} = \mathbf{X}\mathbf{C} + \mathbf{E} + \mathbf{Z},\ \text{diag}(\mathbf{C}) = \mathbf{0} \tag{2.7}$$

Here, $\|\mathbf{E}\|_1$ handles sparse outliers and $\|\mathbf{Z}\|_F$ handles Gaussian noise. The parameters λ_e and λ_z balance the reconstruction error and sparsity. If the data is only affected by Gaussian noise, the term $\mathbf{E}$ can be dropped, reducing the problem to a Lasso-like formulation.

The optimization problem can be solved efficiently using optimization tools. Specifically, the Alternating Direction Method of Multipliers (ADMM) is widely used for this task [2], with linearized variants offering improved convergence properties for ℓ_1-minimization problems [39]. ADMM decomposes the problem into simpler subproblems that can be updated iteratively and is known to be effective for sparse subspace clustering [1]. In this work, ADMM is used to solve the optimization problem.

Once the sparse coefficient matrix $\mathbf{C}$ is obtained, it is often normalized to better handle variations in data magnitude. The columns are normalized as $\mathbf{c}_i \leftarrow \frac{\mathbf{c}_i}{\|\mathbf{c}_i\|_\infty}$. Subsequently, an affinity matrix $\mathbf{W}$ is constructed for spectral clustering:

$$\mathbf{W} = |\mathbf{C}| + |\mathbf{C}|^T \tag{2.8}$$

This symmetrization ensures that nodes i and j are connected if either point participates in the sparse representation of the other [17]. Finally, the segmentation is obtained by applying spectral clustering [19] to the similarity graph defined by $\mathbf{W}$.

In this work, SSC serves as a comparison method with Block Diagonal Representation approaches. The implementation includes both standard inner product-based formulation ($\mathbf{X}^T\mathbf{X}$) and a variant using Gaussian RBF kernel based on Euclidean distance, allowing for investigation of how different similarity measures affect clustering quality. The hyperparameters (sparsity regularization weight (λ), ADMM

penalty parameter (ρ), and affine constraint) are systematically explored across multiple feature extraction methods to identify optimal configurations for image clustering.

2.2.2 Block Diagonal Representation (BDR)

Block Diagonal Representation (BDR) [1] is an alternative approach to SSC that directly enforces block diagonal structure in the coefficient matrix, rather than relying on sparsity to indirectly promote this structure. This direct approach is particularly relevant for image clustering, where clear separation between image categories is expected.

The core of BDR is the k-block diagonal regularizer, denoted as $\|\mathbf{B}\|_{\boxed{k}}$. For an affinity matrix $\mathbf{B} \in \mathbb{R}^{N\times N}$ (where $\mathbf{B} \geq 0$ and $\mathbf{B} = \mathbf{B}^T$), the number of connected components in the corresponding graph is equal to the multiplicity of the zero eigenvalue of its Laplacian matrix $\mathbf{L_B} = \text{Diag}(\mathbf{B1}) - \mathbf{B}$ [15]. Based on this spectral graph theory property, the k-block diagonal regularizer is defined as the sum of the k smallest eigenvalues of the Laplacian:

$$\|\mathbf{B}\|_{\boxed{k}} = \sum_{i=N-k+1}^{N} \lambda_i(\mathbf{L_B}) \tag{2.9}$$

where $\lambda_i(\mathbf{L_B})$ denotes the eigenvalues of $\mathbf{L_B}$ in decreasing order. Minimizing this term enforces the Laplacian to have null eigenvalues, thereby encouraging the formation of k connected components (blocks).

The BDR formulation combines the self-expressiveness property with this structural regularizer. Since enforcing block diagonality directly on the coefficient matrix $\mathbf{Z}$ is difficult due to non-symmetry and negative values, BDR introduces an auxiliary symmetric affinity matrix $\mathbf{B}$. The optimization problem is formulated as:

$$\begin{aligned} \min_{\mathbf{Z},\mathbf{B}} \quad & \frac{1}{2}\|\mathbf{X} - \mathbf{XZ}\|_F^2 + \frac{\lambda}{2}\|\mathbf{Z} - \mathbf{B}\|_F^2 + \gamma\|\mathbf{B}\|_{\boxed{k}} \\ \text{s.t.} \quad & \text{diag}(\mathbf{B}) = \mathbf{0}, \mathbf{B} \geq 0, \mathbf{B} = \mathbf{B}^T \end{aligned} \tag{2.10}$$

where λ controls the coupling between the representation matrix $\mathbf{Z}$ and the block-diagonal affinity matrix $\mathbf{B}$, and γ controls the strength of the block diagonal constraint. The term $\|\mathbf{Z}-\mathbf{B}\|_F^2$ ensures that the learned representation $\mathbf{Z}$ remains close to the strictly block-diagonal matrix $\mathbf{B}$.

This problem is non-convex but can be solved using an alternating minimization strategy. Once the optimization converges, the final clustering is obtained by applying spectral clustering to an affinity matrix. Lu et al. [1] propose two variants

for constructing this affinity matrix:

BDR-B: The affinity matrix is constructed from the block-diagonal matrix $\mathbf{B}$:

$$\mathbf{W} = \frac{|\mathbf{B}| + |\mathbf{B}|^T}{2} \tag{2.11}$$

Since $\mathbf{B}$ is explicitly regularized to have k blocks, this variant yields a clean graph structure that directly reflects the block-diagonal prior.

BDR-Z: The affinity matrix is constructed from the reconstruction coefficient matrix $\mathbf{Z}$:

$$\mathbf{W} = \frac{|\mathbf{Z}| + |\mathbf{Z}|^T}{2} \tag{2.12}$$

Although $\mathbf{B}$ is strictly constrained to be non-negative, symmetric, and block-diagonal, $\mathbf{Z}$ tends to be "close to but denser than" $\mathbf{B}$ [1]. This denser connectivity may offer better robustness when the strict block-diagonal assumption is slightly violated by noise or complex data correlations.

2.3 Spectral Clustering

Spectral clustering is a graph-based clustering technique that leverages the spectral decomposition of graph Laplacian matrices to identify clusters. In the context of this work, spectral clustering serves as the final step in both SSC and BDR methods to obtain the actual cluster assignments from the learned affinity matrices. Unlike traditional clustering methods such as K-means, spectral clustering can handle non-convex cluster shapes and is particularly effective when the data contains a subspace structure [19, 15].

Given an affinity matrix constructed by the sparse representation step (typically symmetrized as $\mathbf{W} = |\mathbf{C}| + |\mathbf{C}|^T$), spectral clustering treats the data as a weighted undirected graph where nodes represent data points and edge weights represent similarities. The goal is to partition this graph into k clusters such that edges within clusters have high weights while edges between clusters have low weights [20].

In this study, spectral clustering is applied uniformly across all methods to ensure a fair comparison. Symmetric normalized Laplacian $\mathbf{L}_{\text{sym}}$ and the row-normalization are used as described in the NJW algorithm [19]. The number of clusters k is assumed to be known prior. By keeping the spectral clustering parameters identical, it has been ensured that observed performance differences are attributable to the quality of the affinity matrices learned by the subspace clustering

algorithms.

2.3.1 Graph Laplacian

The foundation of spectral clustering is in the graph Laplacian matrix. Let $\mathbf{W}$ be a symmetric affinity matrix with non-negative entries. The degree matrix $\mathbf{D}$ is a diagonal matrix defined as $d_{ii} = \sum_{j=1}^{N} w_{ij}$.

The unnormalized graph Laplacian is defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$. While the unnormalized Laplacian has important theoretical properties, normalized Laplacian is preferred for clustering tasks due to their better handling of irregular graph degrees [15]. The symmetric normalized Laplacian is defined as:

$$\mathbf{L}_{\text{sym}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} \tag{2.13}$$

Alternatively, the random walk normalized Laplacian is defined as $\mathbf{L}_{\text{rw}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$. The choice of Laplacian affects the subsequent spectral decomposition. In subspace clustering literature, the symmetric normalized Laplacian (typically adopted by standard spectral clustering algorithms such as the one by Ng et al. [19]) is the standard choice.

2.3.2 Spectral Decomposition and Clustering

A fundamental result in spectral graph theory states that the multiplicity of the zero eigenvalue of the Laplacian equals the number of connected components in the graph [6]. This property is explicitly utilized by the Block Diagonal Representation (BDR) method, which incorporates a k-block diagonal regularizer to enforce the sum of the k smallest eigenvalues to be zero, thereby promoting the formation of k connected components.

The spectral clustering algorithm proceeds as follows:

1. Compute the symmetric normalized Laplacian $\mathbf{L}_{\text{sym}}$ from the affinity matrix $\mathbf{W}$.

2. Compute the k smallest eigenvalues and their corresponding eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ of $\mathbf{L}_{\text{sym}}$.

3. Form the matrix $\mathbf{V} \in \mathbb{R}^{N \times k}$ by stacking these eigenvectors as columns: $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$.

4. Form the normalized matrix $\mathbf{Y} \in \mathbb{R}^{N \times k}$ by normalizing each row of $\mathbf{V}$ to have

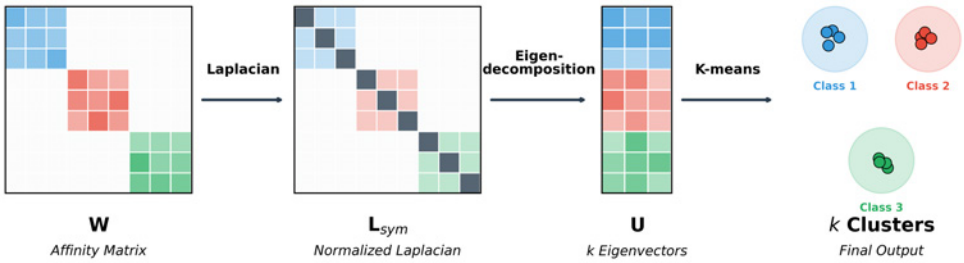


Figure 2.3: Illustration of the spectral clustering pipeline. Starting from the learned affinity matrix $\mathbf{W}$, the normalized Laplacian $\mathbf{L}_{sym}$ is computed. Eigendecomposition yields k eigenvectors corresponding to the smallest eigenvalues, which form the embedding matrix $\mathbf{U}$. Finally, K-means clustering is applied to the rows of $\mathbf{U}$ to obtain the final cluster assignments. The block-diagonal structure in $\mathbf{W}$ leads to well-separated clusters in the eigenvector space.

unit Euclidean length:

$$\mathbf{y}_i = \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|_2} \tag{2.14}$$

This step projects the data points onto the unit hypersphere.

5. Apply K-means clustering on the rows of $\mathbf{Y}$ (treating each row $\mathbf{y}_i$ as a data point in $\mathbb{R}^k$) to obtain k clusters.

The key insight is that the eigenvector space provides a low-dimensional embedding where data points belonging to the same cluster are grouped together, making them easier to separate via K-means [19]. Figure 2.3 illustrates this complete spectral clustering pipeline, showing how the affinity matrix is transformed through Laplacian computation and eigendecomposition into a low-dimensional embedding space where K-means can effectively identify clusters.

2.4 Graph-based Image Clustering

The sparse representation-based subspace clustering methods described in the previous sections (SSC and BDR variants) are fundamentally graph-based approaches to image clustering. In these methods, images are treated as nodes in a weighted graph, where the edge weights are determined by the sparse coefficients learned from the self-expressiveness property [17]. This section discusses how these theoretical concepts are applied specifically to image clustering tasks.

In graph-based image clustering, the main challenge is to construct an appropriate affinity matrix $\mathbf{W}$ that accurately captures the relationships between images.

The affinity matrix defines a similarity graph where images belonging to the same object category should form tightly connected components. The quality of this affinity matrix directly impacts the final clustering performance, as spectral clustering operates on this graph structure to identify the underlying clusters [19, 15].

For image data, the construction of the affinity matrix proceeds in two stages. First, a sparse coefficient matrix is computed using either SSC or BDR formulations. This step enforces the self-expressiveness property, ensuring that each image is represented as a sparse linear combination of other images from the same subspace [17]. Second, the coefficient matrix is processed through thresholding and symmetrization operations to construct the final affinity matrix. For SSC, the affinity matrix is typically constructed as $\mathbf{W} = |\mathbf{C}| + |\mathbf{C}|^T$, where $\mathbf{C}$ is the sparse coefficient matrix [17]. For BDR, the block-diagonal matrix $\mathbf{B}$ or the reconstruction matrix $\mathbf{Z}$ is similarly symmetrized. In both cases, an additional thresholding step with parameter ρ is often applied to remove weak connections and enhance the block-diagonal structure [1].

The effectiveness of graph-based image clustering depends critically on the feature representation used to describe the images. Different feature extraction methods capture different aspects of image content, leading to varying clustering performance. Raw pixel intensities provide the most direct representation but are sensitive to variations in illumination, pose, and background. More sophisticated feature extraction methods, such as local descriptor-based approaches (Bag-of-Visual-Words [3], VLAD [4], Fisher Vectors [5]), can provide more robust representations by encoding local image statistics into a global feature vector. This work systematically evaluates multiple feature extraction methods to identify the most effective representations for subspace clustering of object images.

2.4.1 Feature Extraction

Feature extraction transforms raw images into numerical vectors suitable for subspace clustering algorithms. The choice of feature representation significantly impacts clustering performance, as it determines how images are positioned in the feature space and, consequently, how well the subspace structure can be recovered. Many of these methods rely on extracting local features, such as Scale-Invariant Feature Transform (SIFT) [21], which are then aggregated into a global representation. While recent advances in deep learning have led to learned representations for subspace clustering [34], this work focuses on hand-crafted features to enable systematic analysis of feature properties without the confounding effects of end-to-end learning. Four distinct feature extraction approaches are evaluated: raw pixel in-

tensities, Bag-of-Visual-Words (BoVW), Vector of Locally Aggregated Descriptors (VLAD), and Fisher Vectors (FV). Each method offers different trade-offs between representational power, computational complexity, and robustness to image variations.

Raw Pixel Features

Raw pixel intensity values serve as the most fundamental baseline representation for image clustering. In this approach, input images are resized to a fixed resolution to ensure dimensionality consistency across the dataset. The 2D image matrices are then flattened (vectorized) into 1D vectors $\mathbf{x}_i \in \mathbb{R}^D$, where D is the total number of pixels.

To account for variations in global illumination and contrast, the feature vectors are typically normalized. A common approach is to apply ℓ_2-normalization to project each feature vector onto the unit hypersphere:

$$\mathbf{x}_i \leftarrow \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|_2} \tag{2.15}$$

This normalization step is a critical component of subspace clustering pipelines, ensuring that the similarity between data points is driven by the angular separation of their feature vectors rather than their Euclidean magnitudes [17]. Despite their simplicity, raw pixel representations can be highly effective for datasets exhibiting constrained variability. An example is face clustering under varying illumination, where face images have been theoretically and empirically shown to lie near low-dimensional linear subspaces [40]. However, raw pixel features lack invariance to geometric transformations and are susceptible to background clutter, making them generally more prone to error compared to sophisticated descriptor-based approaches.

Bag-of-Visual-Words (BoVW)

The Bag-of-Visual-Words (BoVW) model, inspired by text retrieval methods, represents an image as a histogram of visual word occurrences [3]. The process involves three key steps: feature detection and description, visual vocabulary construction, and image encoding.

First, robust local descriptors, for this work SIFT [21], are extracted from keypoints detected in the images. These descriptors are designed to be invariant to scale, rotation, and illumination changes. A visual vocabulary is then constructed by clustering a large set of these descriptors using the K-means algorithm [41]. The cluster centers serve as the "visual words" that form the codebook.

Each image is then encoded by assigning its local descriptors to the nearest visual word in the vocabulary (hard assignment) and aggregating them into a frequency histogram $\mathbf{h}_i \in \mathbb{R}^K$, where K is the vocabulary size. Finally, the resulting histograms are typically ℓ_2-normalized to ensure invariance to the total number of keypoints detected in the image:

$$\mathbf{x}_i = \frac{\mathbf{h}_i}{\|\mathbf{h}_i\|_2} \tag{2.16}$$

This representation effectively captures the global distribution of local textures and patterns but discards spatial information.

Vector of Locally Aggregated Descriptors (VLAD)

The Vector of Locally Aggregated Descriptors (VLAD) extends the BoVW method by encoding not only the assignment of descriptors to visual words, but also the residuals (differences) between descriptors and their assigned cluster centers [4]. This approach preserves more fine-grained information about the local feature distribution while maintaining a compact global representation.

Like BoVW, VLAD begins with the construction of a visual vocabulary of K visual words (codebook) via K-means clustering. For each image, local descriptors are extracted and assigned to their nearest visual words. However, instead of entirely counting occurrences, VLAD accumulates the residual vectors. For the k-th visual word, the residual sum is computed as:

$$\mathbf{v}_k = \sum_{\mathbf{x}_i : NN(\mathbf{x}_i)=k} (\mathbf{x}_i - \boldsymbol{\mu}_k) \tag{2.17}$$

where $\mathbf{x}_i$ denotes a local descriptor, $NN(\mathbf{x}_i)$ indicates the nearest visual word, and $\boldsymbol{\mu}_k$ is the k-th cluster center. The final VLAD representation is formed by concatenating all residual vectors $\mathbf{v} = [\mathbf{v}_1^T, \mathbf{v}_2^T, \ldots, \mathbf{v}_K^T]^T \in \mathbb{R}^{K \times d}$, where d is the descriptor dimensionality.

To enhance discriminative power and ensure invariance to the number of features, VLAD vectors typically undergo normalization. Common strategies include intra-normalization (independent ℓ_2-normalization of each $\mathbf{v}_k$) followed by global ℓ_2-normalization of the entire vector [23]. In this work, power normalization is applied (signed square-root normalization) where each element is transformed as $x \leftarrow \text{sign}(x)\sqrt{|x|}$, followed by global ℓ_2-normalization. This two-stage normalization scheme has been shown to improve discriminability by reducing the influence of visual burstiness (the tendency of certain visual words to appear in bursts) [42]. The resulting representation captures both the spatial layout and the distribution

of local features, making it more expressive than BoVW at the cost of higher dimensionality.

Fisher Vector (FV)

The Fisher Vector (FV) extends the BoVW paradigm by modeling the distribution of local descriptors using a probabilistic framework based on Gaussian Mixture Models (GMMs) [5, 43]. Unlike BoVW and VLAD, which rely on hard assignment to a discrete visual vocabulary, Fisher Vector employs a generative model to capture the underlying distribution of visual features, encoding both first-order and second-order statistics.

A GMM with K components is first learned from a training set of local descriptors. The GMM is parameterized by mixture weights $\boldsymbol{\pi} = (\pi_k)$, means $\boldsymbol{\mu} = (\boldsymbol{\mu}_k)$, and covariance matrices $\boldsymbol{\Sigma} = (\boldsymbol{\sigma}_k^2)$ (assuming diagonal covariances for computational efficiency). Given an image represented by a set of local descriptors $\{\mathbf{x}_i\}$, the Fisher Vector encodes the gradient of the log-likelihood with respect to the GMM parameters. Specifically, for each Gaussian component k, the Fisher Vector captures the deviation of the descriptors from the model via gradient vectors with respect to the mean and variance:

$$\mathbf{g}_{\boldsymbol{\mu}_k} = \frac{1}{\sqrt{\pi_k}} \sum_i \gamma_k(\mathbf{x}_i) \frac{\mathbf{x}_i - \boldsymbol{\mu}_k}{\boldsymbol{\sigma}_k} \tag{2.18}$$

$$\mathbf{g}_{\boldsymbol{\sigma}_k} = \frac{1}{\sqrt{2\pi_k}} \sum_i \gamma_k(\mathbf{x}_i) \left[\frac{(\mathbf{x}_i - \boldsymbol{\mu}_k)^2}{\boldsymbol{\sigma}_k^2} - 1 \right] \tag{2.19}$$

where $\gamma_k(\mathbf{x}_i)$ denotes the soft assignment (posterior probability) of descriptor $\mathbf{x}_i$ to the k-th Gaussian component. The final Fisher Vector is formed by concatenating the gradients for all components: $\mathbf{f} = [\mathbf{g}_{\boldsymbol{\mu}_1}^T, \mathbf{g}_{\boldsymbol{\sigma}_1}^T, \ldots, \mathbf{g}_{\boldsymbol{\mu}_K}^T, \mathbf{g}_{\boldsymbol{\sigma}_K}^T]^T \in \mathbb{R}^{2Kd}$.

In this work, the "improved" Fisher Vector formulation [44] is applied, which incorporates power normalization (signed square-root) and ℓ_2-normalization to enhance discriminability and reduce the influence of burstiness. This representation is highly expressive, capturing detailed distributional information about local features [43].

Chapter 3

Methodology

This chapter details the methodology used to perform image clustering using sparse representation based subspace clustering techniques. The primary objective is to systematically compare Sparse Subspace Clustering (SSC) [17] and Block Diagonal Representation (BDR) [1] methods across different feature extraction approaches and similarity measures, identifying the most effective combinations for accurate object categorization. While other subspace clustering methods such as Low-Rank Representation (LRR) [18] exist in the literature, this work focuses on SSC and BDR due to their sparsity-promoting properties and demonstrated effectiveness in image segmentation tasks.

The proposed approach treats images as high dimensional data points and utilizes the self-expressiveness property of subspace clustering [17], where each data point can be represented as a linear combination of other points from the same subspace. Various feature extraction techniques are systematically explored, including raw pixel features, Bag-of-Visual-Words (BoVW) [3], Vector of Locally Aggregated Descriptors (VLAD) [4], and Fisher Vectors [5], to transform images into discriminative representations suitable for subspace clustering. Additionally, the impact of different similarity measures, linear kernel (inner product) and Gaussian RBF kernel based on Euclidean distance, on clustering performance is investigated.

The methodology is divided into seven main sections: dataset description providing details about the image data used for evaluation, proposed pipeline overview presenting the complete system architecture, feature extraction describing the implementation of each feature representation method, subspace clustering implementation detailing the SSC and BDR algorithms, similarity measures explaining the kernel functions employed, evaluation metrics defining the performance measures used, and implementation details covering the software and hardware specifications.

Table 3.1: Summary of the primary dataset characteristics.

Property	**Value**
Total images	533
Number of object categories	16
Image resolution	224×224 pixels
Color format	RGB
Background	Clean (white and black)
Acquisition conditions	Controlled

3.1 Dataset Design

This work utilizes two datasets to evaluate the robustness of subspace clustering methods under both controlled and real-world conditions.

3.1.1 Primary Dataset

The primary dataset consists of 533 images capturing 16 objects. All images were acquired under controlled conditions with consistent lighting, similar camera distance, and clean white and black background. This controlled setting ensures that variations between images of the same object are minimal, establishing an ideal baseline for clustering algorithm evaluation.

Each image has been resized to 224×224 pixels using Lanczos-3 interpolation to ensure uniform input dimensions across the pipeline. The 16 object categories represent common everyday items, providing sufficient visual diversity for meaningful clustering evaluation while maintaining a manageable scale for systematic parameter exploration. Table 3.1 summarizes the dataset characteristics.

3.1.2 Real-World Dataset

To evaluate robustness against real-world domain shifts, a second dataset was collected containing the same 16 object categories captured in uncontrolled natural environments. This dataset comprises 419 images photographed in various locations with varying backgrounds, lighting conditions, camera distances, and viewing angles.

Unlike the controlled dataset where objects are centered and isolated, real-world images contain background clutter and contextual elements. To ensure fair comparison with the primary dataset, a manual Region of Interest (ROI) selection and

(a) Controlled Dataset: Clean white-black background, consistent lighting

(b) Real-World Dataset: Variable backgrounds, natural lighting, clutter

Figure 3.1: Comparison of the same object categories captured under controlled conditions (top row) versus real-world conditions (bottom row). The controlled images feature clean white-black backgrounds and consistent lighting, while the real-world images exhibit background clutter, varying illumination and occlusions.

cropping process was applied to isolate the target objects from their surroundings. This preprocessing step, illustrated in Chapter 4, removes background interference while preserving the natural appearance variations introduced by real-world capture conditions.

The real-world dataset is used exclusively in Experimental Setting 1 (Section 4.3) to systematically evaluate how clustering performance degrades as controlled laboratory images are progressively replaced with their real-world counterparts. Figure 3.1 illustrates the visual difference between corresponding images from both datasets.

3.2 Proposed Pipeline Overview

The proposed image clustering system follows a modular pipeline architecture consisting of five main stages: data loading and preprocessing, feature extraction, sparse representation-based subspace clustering, spectral clustering, and performance evaluation, as illustrated in Figure 3.2. This modular design allows for systematic comparison of different feature extraction methods and clustering algorithms while maintaining consistent evaluation conditions.

3.2.1 Pipeline Architecture

The complete pipeline operates as follows. First, images are loaded from the dataset and preprocessed to ensure uniform dimensions and format. Second, one of four feature extraction methods (raw pixel, BoVW, VLAD, or Fisher Vector) is applied to transform images into high dimensional feature vectors. Third, a similarity measure is selected, either the linear kernel (inner product) or the Gaussian RBF kernel based on Euclidean distance. Fourth, the feature matrix along with the selected similarity measure is passed to either the SSC [17] or BDR [1] solver, which computes the sparse representation coefficient matrix. Fifth, an affinity matrix is constructed from the coefficient matrix and spectral clustering [19] is performed to obtain the final cluster assignments. Finally, the clustering results are evaluated against ground truth labels using accuracy and F1-score metrics.

The pipeline is designed with a caching mechanism for feature extraction, allowing repeated experiments with different clustering parameters without recomputing features. Additionally, the system supports grid search over hyperparameters, enabling systematic exploration of optimal configurations for each feature-clustering combination.

3.2.2 Data Flow

The data flow through the pipeline can be summarized as:

1. **Input:** Raw images from the dataset directory, organized by object class
2. **Preprocessing:** Images are resized to a fixed resolution and converted to grayscale if necessary
3. **Feature Extraction:** Images are transformed into feature vectors $\mathbf{x}_i \in \mathbb{R}^D$, forming the data matrix $\mathbf{X} \in \mathbb{R}^{D \times N}$
4. **Similarity Measure:** A kernel function is selected—either the linear kernel ($\mathbf{X}^T\mathbf{X}$) or the Gaussian RBF kernel based on Euclidean distance
5. **Subspace Clustering:** The coefficient matrix $\mathbf{C}$ (or $\mathbf{Z}$ for BDR) is computed via convex optimization using ADMM [2]
6. **Affinity Construction:** The affinity matrix $\mathbf{W} = |\mathbf{C}| + |\mathbf{C}|^T$ is constructed and thresholded [17]
7. **Spectral Clustering:** Eigendecomposition of the graph Laplacian followed by K-means clustering [19]

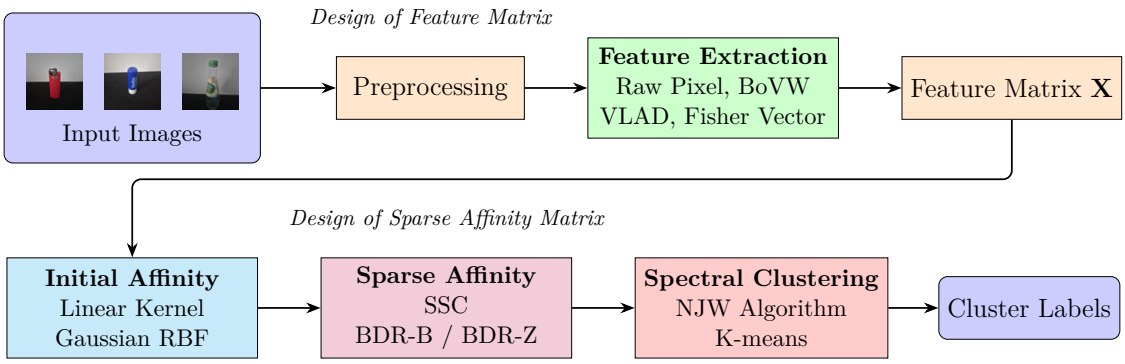


Figure 3.2: Block diagram of the proposed image clustering pipeline. The pipeline consists of three main stages: (1) Feature Matrix Design, where input images are preprocessed and features are extracted; (2) Sparse Affinity Matrix Design, where an initial affinity is computed and sparse representation techniques (SSC or BDR) are applied; (3) Spectral Clustering, where the final cluster labels are obtained.

8. **Output:** Cluster labels for each image, along with accuracy and F1-score metrics

3.3 The Design of Feature Matrix

This section describes the design of the feature matrix $\mathbf{X} \in \mathbb{R}^{D \times N}$, which forms the input to the sparse affinity matrix construction stage. The process involves two main steps: preprocessing the raw images and extracting discriminative feature representations. The theoretical foundations of these methods were presented in Chapter 2; here, the focus is on the specific parameters and implementation choices.

3.3.1 Image Preprocessing

Prior to feature extraction, all images undergo a standardized preprocessing pipeline to ensure consistency across the dataset. The preprocessing steps are as follows:

1. **Resizing:** All images are resized to a fixed resolution of 224×224 pixels using Lanczos-3 interpolation, which provides high-quality anti-aliased results.

2. **Color Conversion:** For methods requiring grayscale input (SIFT-based methods), color images are converted to grayscale using the standard luminance formula: $Y = 0.299R + 0.587G + 0.114B$.

3. **Data Type Normalization:** Pixel values are normalized to the range $[0, 1]$ for numerical stability during feature computation.

The choice of 224×224 resolution balances computational efficiency with sufficient detail preservation for feature extraction. This resolution is consistent with common practices in image classification literature.

3.3.2 Feature Extraction

This subsection describes the four feature extraction methods employed in this work. Each method transforms preprocessed images into fixed-length feature vectors that form the columns of the data matrix $\mathbf{X}$.

Raw Pixel Features

The simplest feature representation treats the image directly as a high-dimensional vector. For each image, the preprocessing steps are followed by:

1. **Vectorization:** The 224×224 grayscale image is flattened into a column vector $\mathbf{x}_i \in \mathbb{R}^{50176}$.
2. **Normalization:** Each feature vector is ℓ_2-normalized: $\mathbf{x}_i \leftarrow \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|_2}$

The resulting feature dimension is $D = 224 \times 224 = 50{,}176$. Despite its simplicity, this representation can be effective for datasets with limited intra-class variability, such as face images under varying illumination [40].

Bag-of-Visual-Words (BoVW)

The Bag-of-Visual-Words (BoVW) implementation follows the standard pipeline described in Chapter 2 and illustrated in Figure 3.3, with the following specific parameters:

1. **Local Feature Detection:** SIFT keypoints are detected at multiple scales using the MATLAB `detectSIFTFeatures` function. SIFT descriptors are chosen for their proven robustness to scale, rotation, and illumination changes [45]. The 80% strongest features (based on response magnitude) are retained to reduce noise from weak detections.
2. **Descriptor Extraction:** 128-dimensional SIFT descriptors are extracted at each keypoint location.
3. **Visual Vocabulary Construction:** K-means clustering [46] is applied to all training descriptors to form a vocabulary of $K = 800$ visual words. The vocabulary is trained only on the training set to prevent information leakage.

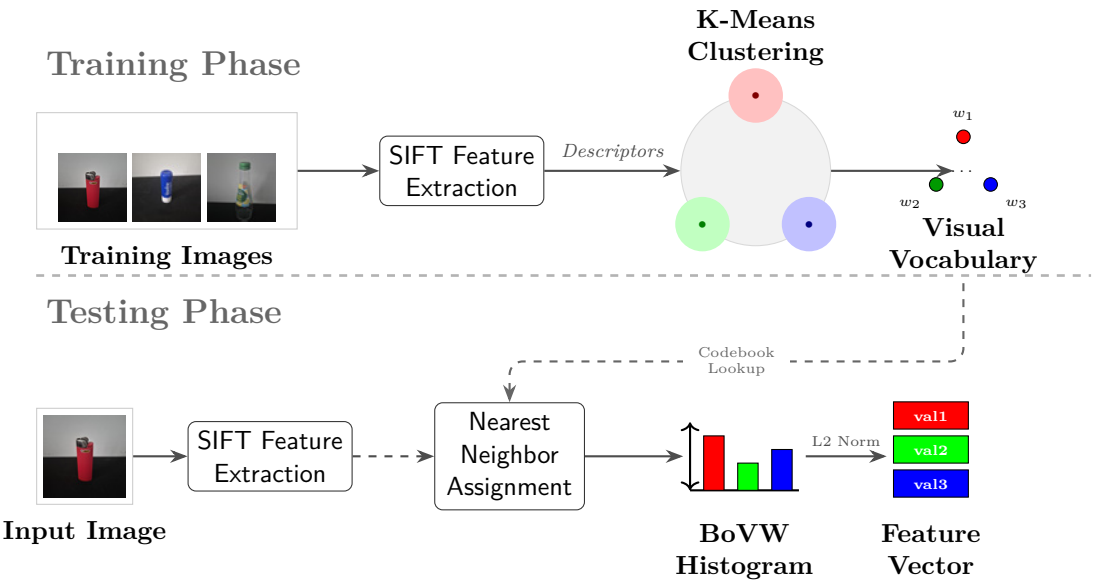


Figure 3.3: Visual illustration of the Bag-of-Visual-Words (BoVW) pipeline (adapted from [47]). The process consists of two phases: **Training Phase** (top), where descriptors extracted from training images are clustered to form a visual vocabulary (codebook); and **Testing Phase** (bottom), where features from a new image are assigned to the nearest visual words to construct a frequency histogram, forming the final feature vector.

4. **Histogram Encoding:** Each image is represented as a histogram of visual word occurrences. Hard assignment is used, where each descriptor is assigned to its nearest visual word.

5. **Normalization:** The histogram is ℓ_2-normalized to produce the final feature vector $\mathbf{x}_i \in \mathbb{R}^{800}$.

The specific parameters used for BoVW in our experiments, such as vocabulary size and feature selection ratio, are provided in the experimental evaluation (Chapter 4).

Vector of Locally Aggregated Descriptors (VLAD)

The Vector of Locally Aggregated Descriptors (VLAD) implementation extends BoVW by encoding residual information [4], as illustrated in Figure 3.4. The following parameters are used:

1. **Dense SIFT Extraction:** Unlike BoVW, VLAD uses dense SIFT descriptors extracted on a regular grid across the image, providing more comprehensive coverage.

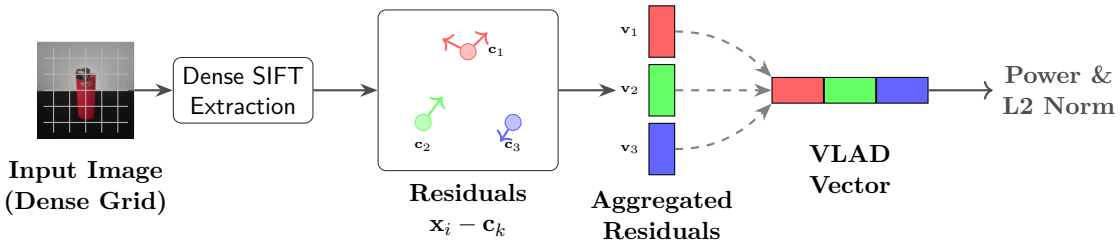


Figure 3.4: Visual illustration of the VLAD pipeline implementation. Dense SIFT descriptors are extracted from the input image on a grid. Each descriptor is assigned to the nearest cluster center ($\mathbf{c}_k$), and the residuals ($\mathbf{x}_i - \mathbf{c}_k$) are accumulated to form sub-vectors ($\mathbf{v}_k$). These are concatenated and normalized to produce the final VLAD representation.

2. **Codebook Construction:** K-means clustering [46] is applied to a subset of up to 200,000 descriptors to form a codebook of $K = 64$ visual words with 10 replicates for robustness.

3. **Residual Encoding:** For each descriptor, the residual (difference from assigned cluster center) is computed and accumulated per cluster.

4. **Normalization:** The aggregated residual vectors are concatenated to form a single vector. Power normalization (signed square-root) is then applied element-wise, followed by global ℓ_2-normalization [42].

The final VLAD feature dimension depends on the codebook size K and descriptor dimension d. Specific implementation parameters are detailed in Chapter 4.

Fisher Vectors (FV)

The Fisher Vector (FV) implementation uses the VLFeat library [48] with the "Improved" formulation [44]. The implementation details are:

1. **Dense SIFT Extraction:** Same as VLAD, dense SIFT descriptors are extracted on a regular grid.

2. **GMM Training:** A Gaussian Mixture Model with $K = 64$ components is trained on a subset of up to 200,000 descriptors using the Expectation-Maximization (EM) algorithm [49]. The EM algorithm runs for a maximum of 500 iterations with 3 random initializations.

3. **Fisher Vector Encoding:** For each image, the gradients with respect to GMM means and variances are computed and concatenated.

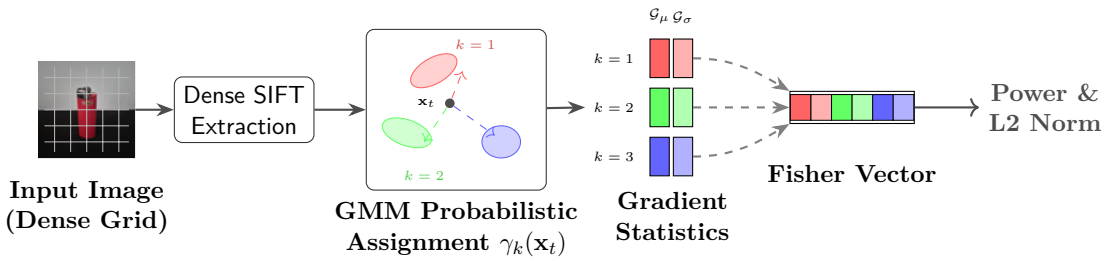


Figure 3.5: Visual illustration of the Fisher Vector (FV) pipeline implementation. Dense SIFT descriptors are extracted and modeled using a Gaussian Mixture Model (GMM). First-order ($\mathcal{G}_\mu$) and second-order ($\mathcal{G}_\sigma$) gradient statistics are computed for each GMM component, concatenated, and normalized to form the high-dimensional Fisher Vector.

4. **Improved FV Normalization:** Power normalization (signed square-root) and ℓ_2-normalization are applied [44].

The resulting Fisher Vector encodes both first-order and second-order statistics, leading to a high-dimensional representation, as illustrated in Figure 3.5. Detailed configuration settings are provided in Chapter 4.

3.4 The Design of Sparse Affinity Matrix

This section describes the construction of the sparse affinity matrix, which is the core component of the subspace clustering pipeline. The process involves two main steps: computing an initial affinity matrix using similarity measures, and then applying sparse representation techniques to obtain the final sparse affinity matrix, as illustrated in Figure 3.6.

3.4.1 Initial Affinity Matrix Construction

The initial affinity matrix captures pairwise similarities between data points. Two similarity measures are systematically compared in this work.

Affinity Matrix Construction using Linear Kernel (Inner Product)

The standard formulation uses the linear kernel (inner product) to measure similarity between feature vectors:

$$\mathbf{G} = \mathbf{X}^T\mathbf{X} \tag{3.1}$$

where $g_{ij} = \mathbf{x}_i^T\mathbf{x}_j$ represents the similarity between data points i and j. When features are ℓ_2-normalized (as in all feature extraction methods used in this work),

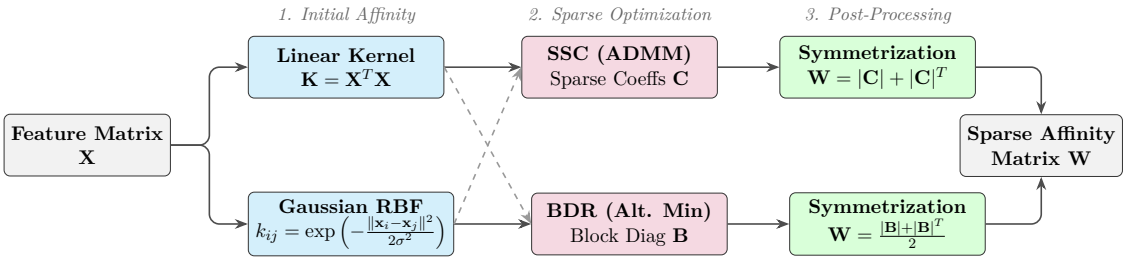


Figure 3.6: Pipeline for Sparse Affinity Matrix construction. The process begins with the computation of an initial kernel matrix $\mathbf{K}$ using either a Linear or Gaussian RBF kernel. This initialized similarity is then refined using sparse representation techniques, Sparse Subspace Clustering (SSC) or Block Diagonal Representation (BDR), to discover the underlying subspace structures. Dashed arrows indicate that either kernel can be paired with either clustering method. Finally, the learned coefficient matrix is symmetrized to produce the final sparse affinity matrix $\mathbf{W}$.

this reduces to the cosine similarity.

In the SSC optimization, the term $\mathbf{X}^T\mathbf{X}$ appears in the Z-update step:

$$\mathbf{Z}^{(k+1)} = (\lambda \mathbf{X}^T\mathbf{X} + \rho \mathbf{I})^{-1}(\cdots) \tag{3.2}$$

Similarly, in BDR, the Gram matrix appears in the reconstruction term $\|\mathbf{X} - \mathbf{X}\mathbf{Z}\|_F^2$, which expands to involve $\mathbf{X}^T\mathbf{X}$.

Affinity Matrix Construction using Gaussian RBF Kernel

To capture non-linear relationships between data points, a Gaussian Radial Basis Function (RBF) kernel is employed as an alternative similarity measure [38]:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right) \tag{3.3}$$

where σ is the kernel bandwidth parameter controlling the locality of the similarity measure.

In the Euclidean variants of SSC and BDR, the Gram matrix $\mathbf{X}^T\mathbf{X}$ is replaced with the kernel matrix $\mathbf{K}$:

$$k_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right) \tag{3.4}$$

This substitution allows the algorithms to discover non-linear subspace structures that may not be apparent in the original feature space. The kernel bandwidth σ is treated as a hyperparameter that controls the locality of the similarity mea-

sure. Small σ values produce highly localized similarities, while larger values allow connections between more distant points.

Implementation Considerations

The Euclidean distance matrix is computed efficiently using the identity:

$$\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 = \|\mathbf{x}_i\|_2^2 + \|\mathbf{x}_j\|_2^2 - 2\mathbf{x}_i^T\mathbf{x}_j \tag{3.5}$$

This allows leveraging optimized matrix operations rather than explicit pairwise computations. For ℓ_2-normalized features, the first two terms equal 1, simplifying to:

$$\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 = 2(1 - \mathbf{x}_i^T\mathbf{x}_j) \tag{3.6}$$

Both the linear kernel (standard) and Gaussian RBF kernel (Euclidean) variants are systematically evaluated for each feature extraction method, enabling analysis of how similarity measure choice affects clustering performance across different feature representations.

3.4.2 Sparse Affinity Matrix Construction

Once the initial affinity matrix is computed, sparse representation techniques are applied to construct a sparse affinity matrix that reveals the underlying subspace structure. This subsection describes the two methods implemented: Sparse Subspace Clustering (SSC) and Block Diagonal Representation (BDR).

Sparse Subspace Clustering (SSC)

The SSC implementation follows the ADMM-based formulation described by Elhamifar and Vidal [17], with additional optimizations for computational efficiency. The optimization problem solved is:

$$\min_{\mathbf{C}} \frac{\lambda}{2}\|\mathbf{X} - \mathbf{X}\mathbf{C}\|_F^2 + \|\mathbf{C}\|_1 \quad \text{s.t.} \quad \text{diag}(\mathbf{C}) = \mathbf{0} \tag{3.7}$$

where $\mathbf{X} \in \mathbb{R}^{D\times N}$ is the feature matrix with each column representing a data point, and λ is the regularization parameter balancing reconstruction fidelity and sparsity.

ADMM Optimization

The Alternating Direction Method of Multipliers (ADMM) [2] reformulates the problem by introducing an auxiliary variable $\mathbf{Z}$:

$$\min_{\mathbf{C},\mathbf{Z}} \frac{\lambda}{2}\|\mathbf{X} - \mathbf{X}\mathbf{Z}\|_F^2 + \|\mathbf{C}\|_1 \quad \text{s.t.} \quad \mathbf{Z} = \mathbf{C},\ \text{diag}(\mathbf{C}) = \mathbf{0} \tag{3.8}$$

The augmented Lagrangian is formed with penalty parameter ρ, and the algorithm alternates between updating $\mathbf{Z}$, $\mathbf{C}$, and the dual variable $\mathbf{\Lambda}$:

1. **Z-update:** Solve a linear system involving the kernel matrix $\mathbf{K}$:

$$\mathbf{Z}^{(k+1)} = (\lambda\mathbf{K} + \rho\mathbf{I})^{-1}(\lambda\mathbf{K} + \rho\mathbf{C}^{(k)} - \mathbf{\Lambda}^{(k)}) \tag{3.9}$$

   where $\mathbf{K} = \mathbf{X}^T\mathbf{X}$ for the linear kernel variant, or $k_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/2\sigma^2)$ for the Gaussian RBF kernel variant. The matrix inversion is precomputed via Cholesky factorization, reducing per-iteration complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2)$.

2. **C-update:** Apply element-wise soft-thresholding followed by diagonal zeroing:

$$\mathbf{C}^{(k+1)} = \mathcal{S}_{1/\rho}(\mathbf{Z}^{(k+1)} + \mathbf{\Lambda}^{(k)}/\rho), \quad c_{ii} = 0 \tag{3.10}$$

   where $\mathcal{S}_\tau(x) = \text{sign}(x) \cdot \max(0, |x| - \tau)$ is the soft-thresholding operator.

3. **Dual update:**

$$\mathbf{\Lambda}^{(k+1)} = \mathbf{\Lambda}^{(k)} + \rho(\mathbf{Z}^{(k+1)} - \mathbf{C}^{(k+1)}) \tag{3.11}$$

The algorithm terminates when the primal residual $\|\mathbf{Z} - \mathbf{C}\|_\infty$ falls below the convergence tolerance or the maximum number of iterations is reached.

Parameter Configuration

Table 3.2 summarizes the SSC parameters used in this work. The regularization parameter λ controls the trade-off between data fidelity and sparsity; larger values encourage sparser solutions. The ADMM penalty parameter ρ affects convergence speed but not the optimal solution. The affine constraint option enforces $\mathbf{1}^T\mathbf{C} = \mathbf{1}^T$, which is appropriate when data points lie in affine rather than linear subspaces.

The optimal λ value is feature-dependent due to varying data characteristics across feature extraction methods. A grid search is performed for each feature

Table 3.2: SSC implementation parameters.

Parameter	**Symbol**	**Values Tested**
Sparsity regularization	λ	$\{115, 120, 125\}$
ADMM penalty / Threshold	ρ	$\{0.55, 0.65, 0.75\}$
Affine constraint	—	$\{\text{false}, \text{true}\}$
Maximum iterations	—	1000
Convergence tolerance	—	10^{-4}

type to identify the best-performing parameter configuration based on clustering accuracy.

Affinity Matrix Construction

Once the sparse coefficient matrix $\mathbf{C}$ is obtained, the affinity matrix for spectral clustering is constructed through thresholding and symmetrization:

1. **Thresholding:** Small coefficients are removed by applying a threshold based on the parameter ρ:

$$\tilde{c}_{ij} = \begin{cases} c_{ij} & \text{if } |c_{ij}| > \rho \cdot \max_j |c_{ij}| \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

   This step enhances the block-diagonal structure by eliminating weak connections.

2. **Symmetrization:** The affinity matrix is constructed as:

$$\mathbf{W} = |\tilde{\mathbf{C}}| + |\tilde{\mathbf{C}}|^T \tag{3.13}$$

   This ensures that nodes i and j are connected if either point participates in the sparse representation of the other [17].

The resulting affinity matrix $\mathbf{W}$ is then passed to the spectral clustering algorithm (Section 3.5) to obtain the final cluster assignments. Algorithm 1 summarizes the complete SSC procedure.

Algorithm 1 Sparse Subspace Clustering via ADMM

Require: Data matrix $\mathbf{X} \in \mathbb{R}^{D \times N}$, number of clusters k, parameters λ, ρ, kernel type

Ensure: Cluster assignments for each data point

if Linear Kernel **then**
  $\mathbf{K} \leftarrow \mathbf{X}^T\mathbf{X}$ {Gram matrix}
else
  $d_{ij} \leftarrow \|\mathbf{x}_i - \mathbf{x}_j\|_2^2$ {Squared Euclidean distances}
  $\mathbf{K} \leftarrow \exp(-\mathbf{D}/(2\sigma^2))$ {Gaussian RBF kernel}
end if
Precompute $\mathbf{A} = (\lambda\mathbf{K} + \rho\mathbf{I})^{-1}$ via Cholesky factorization
Initialize $\mathbf{C} \leftarrow \mathbf{0}$, $\mathbf{Z} \leftarrow \mathbf{0}$, $\mathbf{\Lambda} \leftarrow \mathbf{0}$
repeat
  $\mathbf{Z} \leftarrow \mathbf{A}(\lambda\mathbf{K} + \rho\mathbf{C} - \mathbf{\Lambda})$ {Z-update}
  $\mathbf{C} \leftarrow \mathcal{S}_{1/\rho}(\mathbf{Z} + \mathbf{\Lambda}/\rho)$ {Soft-thresholding}
  $\text{diag}(\mathbf{C}) \leftarrow \mathbf{0}$ {Zero diagonal}
  $\mathbf{\Lambda} \leftarrow \mathbf{\Lambda} + \rho(\mathbf{Z} - \mathbf{C})$ {Dual update}
until $\|\mathbf{Z} - \mathbf{C}\|_\infty <$ tol or max iterations reached
$\tilde{\mathbf{C}} \leftarrow \text{Threshold}(\mathbf{C}, \rho)$ {Remove weak connections}
$\mathbf{W} \leftarrow |\tilde{\mathbf{C}}| + |\tilde{\mathbf{C}}|^T$ {Symmetrize}
Apply spectral clustering to $\mathbf{W}$ with k clusters
return Cluster labels

Block Diagonal Representation (BDR)

The BDR implementation follows the alternating minimization algorithm proposed by Lu et al. [1]. The optimization problem solved is:

$$\min_{\mathbf{Z},\mathbf{B},\mathbf{W}} \frac{1}{2}\|\mathbf{X} - \mathbf{X}\mathbf{Z}\|_F^2 + \frac{\lambda}{2}\|\mathbf{Z} - \mathbf{B}\|_F^2 + \gamma\langle \text{Diag}(\mathbf{B}\mathbf{1}) - \mathbf{B}, \mathbf{W}\rangle \tag{3.14}$$

subject to $\text{diag}(\mathbf{B}) = \mathbf{0}$, $\mathbf{B} \geq 0$, $\mathbf{B} = \mathbf{B}^T$, and $0 \leq \mathbf{W} \leq \mathbf{I}$, $\text{Tr}(\mathbf{W}) = k$.

Alternating Minimization

The algorithm alternates between updating three variables:

1. **Z-update:** Solve a linear system involving the kernel matrix $\mathbf{K}$:

$$\mathbf{Z}^{(k+1)} = (\mathbf{K} + \lambda\mathbf{I})^{-1}(\mathbf{K} + \lambda\mathbf{B}^{(k)}) \tag{3.15}$$

   where $\mathbf{K} = \mathbf{X}^T\mathbf{X}$ for the linear kernel variant, or $k_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/2\sigma^2)$ for the Gaussian RBF kernel variant. The matrix inversion is precomputed once and reused across iterations.

Table 3.3: BDR implementation parameters.

Parameter	**Symbol**	**Values Tested**
Coupling parameter	λ	$\{15, 17, 19\}$
Block diagonal regularization	γ	$\{0.95, 1.0\}$
Threshold	ρ	$\{0.35, 0.4, 0.45\}$
Maximum iterations	—	1000
Convergence tolerance	—	10^{-3}

2. **B-update:** Update the block-diagonal affinity matrix:

$$\mathbf{B}^{(k+1)} = \mathcal{P}_+ \left(\frac{\mathbf{Z}^{(k+1)} + (\mathbf{Z}^{(k+1)})^T}{2} - \frac{\gamma}{\lambda}(\text{Diag}(\mathbf{W}^{(k)}\mathbf{1}) - \mathbf{W}^{(k)}) \right) \tag{3.16}$$

   where $\mathcal{P}_+$ projects onto the non-negative cone and enforces zero diagonal.

3. **W-update:** Update the spectral constraint matrix via eigendecomposition of the Laplacian $\mathbf{L_B} = \text{Diag}(\mathbf{B1}) - \mathbf{B}$:

$$\mathbf{W}^{(k+1)} = \mathbf{V}_k \mathbf{V}_k^T \tag{3.17}$$

   where $\mathbf{V}_k$ contains the k eigenvectors corresponding to the k smallest eigenvalues of $\mathbf{L_B}$.

The algorithm terminates when $\max(\|\mathbf{Z}^{(k+1)} - \mathbf{Z}^{(k)}\|_\infty, \|\mathbf{B}^{(k+1)} - \mathbf{B}^{(k)}\|_\infty) < 10^{-3}$ or the maximum iteration count (1000) is reached.

BDR-B and BDR-Z Variants

Two variants are implemented based on which matrix is used for spectral clustering:

- **BDR-B:** Uses the block-diagonal matrix $\mathbf{B}$ for clustering. The affinity matrix is $\mathbf{W} = (|\mathbf{B}| + |\mathbf{B}|^T)/2$.
- **BDR-Z:** Uses the reconstruction coefficient matrix $\mathbf{Z}$ for clustering. The affinity matrix is $\mathbf{W} = (|\mathbf{Z}| + |\mathbf{Z}|^T)/2$.

As noted by Lu et al. [1], $\mathbf{Z}$ tends to be denser than $\mathbf{B}$, which may provide better robustness when the strict block-diagonal assumption is violated by noise.

Algorithm 2 Block Diagonal Representation via Alternating Minimization

Require: Data matrix $\mathbf{X} \in \mathbb{R}^{D \times N}$, number of clusters k, parameters λ, γ, kernel type

Ensure: Cluster assignments for each data point

if Linear Kernel **then**
  $\mathbf{K} \leftarrow \mathbf{X}^T\mathbf{X}$ {Gram matrix}
else
  $d_{ij} \leftarrow \|\mathbf{x}_i - \mathbf{x}_j\|_2^2$ {Squared Euclidean distances}
  $\mathbf{K} \leftarrow \exp(-\mathbf{D}/(2\sigma^2))$ {Gaussian RBF kernel}
end if
Precompute $\mathbf{A} = (\mathbf{K} + \lambda\mathbf{I})^{-1}$
Initialize $\mathbf{Z} \leftarrow \mathbf{0}$, $\mathbf{B} \leftarrow \mathbf{0}$, $\mathbf{W} \leftarrow \mathbf{0}$
repeat
  $\mathbf{Z} \leftarrow \mathbf{A}(\mathbf{K} + \lambda\mathbf{B})$ {Z-update}
  $\mathbf{B} \leftarrow \mathbf{Z} - \frac{\gamma}{\lambda}(\text{Diag}(\mathbf{W1}) - \mathbf{W})$
  $\mathbf{B} \leftarrow \max(0, (\mathbf{B} + \mathbf{B}^T)/2)$ {Symmetrize and project}
  $\text{diag}(\mathbf{B}) \leftarrow \mathbf{0}$ {Zero diagonal}
  $\mathbf{L_B} \leftarrow \text{Diag}(\mathbf{B1}) - \mathbf{B}$ {Graph Laplacian}
  $\mathbf{V}_k \leftarrow k$ smallest eigenvectors of $\mathbf{L_B}$
  $\mathbf{W} \leftarrow \mathbf{V}_k\mathbf{V}_k^T$ {W-update}
until convergence or max iterations reached
$\mathbf{W}_{\text{aff}} \leftarrow (|\mathbf{B}| + |\mathbf{B}|^T)/2$ {BDR-B variant}
Apply spectral clustering to $\mathbf{W}_{\text{aff}}$ with k clusters
return Cluster labels

Parameter Configuration

Table 3.3 summarizes the BDR parameters used in this work.

Algorithm 2 summarizes the complete BDR procedure.

3.5 Spectral Clustering

Once the affinity matrix $\mathbf{W}$ is constructed (from either SSC or BDR), spectral clustering is applied following the Ng-Jordan-Weiss (NJW) algorithm [19], as visually summarized in Figure 3.7:

1. **Degree Matrix:** Compute $\mathbf{D} = \text{Diag}(\mathbf{W1})$.
2. **Normalized Laplacian:** Compute $\mathbf{L}_{\text{sym}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$.
3. **Eigendecomposition:** Extract the k eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ corresponding to the k smallest eigenvalues.
4. **Embedding:** Form $\mathbf{V} \in \mathbb{R}^{N \times k}$ and row-normalize: $\mathbf{y}_i = \mathbf{v}_i/\|\mathbf{v}_i\|_2$.

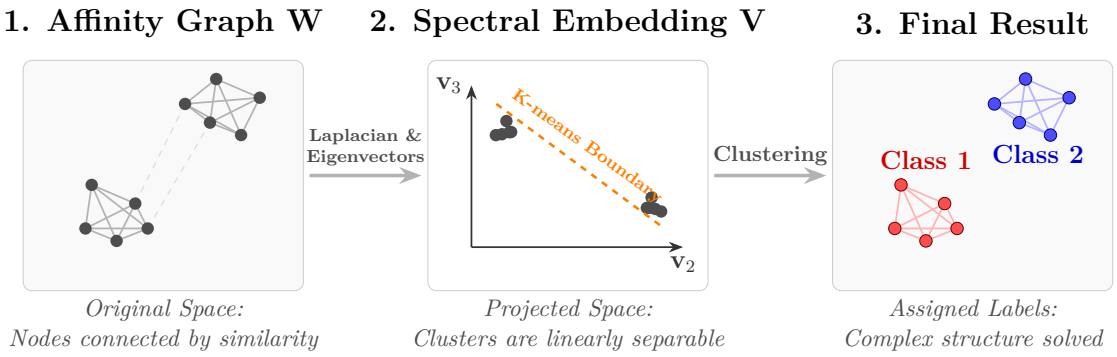


Figure 3.7: Conceptual visualization of Spectral Clustering. (1) An affinity graph $\mathbf{W}$ is constructed where edges represent similarity. (2) The data is projected into a lower-dimensional eigen-space (spectral embedding) using the eigenvectors of the graph Laplacian, making the clusters linearly separable. (3) K-means partitions the data in this new space, revealing the underlying groups in the original domain.

Algorithm 3 Spectral Clustering (NJW Algorithm)

Require: Affinity matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$, number of clusters k
Ensure: Cluster assignments for N vertices
1: Construct degree matrix $\mathbf{D} = \text{Diag}(\mathbf{W1})$
2: Compute normalized Laplacian $\mathbf{L}_{\text{sym}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$
3: Find eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ corresponding to k smallest eigenvalues
4: Form matrix $\mathbf{V} \in \mathbb{R}^{N \times k}$ with columns $\mathbf{v}_1, \ldots, \mathbf{v}_k$
5: Normalize rows: $\mathbf{y}_i \leftarrow \mathbf{v}_i / \|\mathbf{v}_i\|_2$ for $i = 1, \ldots, N$
6: Apply K-means clustering to rows $\{\mathbf{y}_i\}_{i=1}^{N}$
7: Assign vertex i to cluster c if row $\mathbf{y}_i$ was assigned to cluster c
8: **return** Cluster labels

5. **K-means:** Apply K-means clustering [46] to the rows of $\mathbf{Y}$ to obtain cluster assignments.

The number of clusters k is set equal to the known number of object classes in the dataset. K-means is run with 10 random initializations to mitigate sensitivity to initialization. Algorithm 3 summarizes the spectral clustering procedure.

Chapter 4

Experimental Evaluation

This chapter presents the experimental evaluation of the sparse subspace clustering methodologies proposed in Chapter 3 for object clustering. The primary objective is to systematically assess the performance of Sparse Subspace Clustering (SSC) [17] and Block Diagonal Representation (BDR) [1] algorithms under varying data conditions to identify the most effective combinations of feature extraction techniques, similarity measures, graph construction methods and parameters. Two experimental settings are designed to evaluate the robustness and generalization capability of the proposed pipeline. The first setting investigates the impact of replacing controlled-environment images with real-world images exhibiting natural background variations and imaging artifacts. A manual cropping of the images is performed to remove the background variations, focusing only on object itself. The second setting examines the algorithms' resilience against three types of synthetic noise: Gaussian, speckle and occlusion noise. Together, these experiments provide comprehensive insights into the practical applicability of subspace clustering methods for image clustering tasks under realistic conditions.

4.1 Experimental Setup

This section describes the software and hardware environment used for the experiments, as well as practical implementation considerations.

4.1.1 Software Environment

The experiments were implemented in MATLAB R2025b with the following toolboxes:

- **Computer Vision Toolbox:** Used for SIFT feature extraction and Bag-of-Visual-Words functionality.

Table 4.1: Bag-of-Visual-Words implementation parameters.

Parameter	**Value**
Vocabulary size (K)	800
Strongest features ratio	0.8
Descriptor dimension	128 (SIFT)
Feature dimension	800
Normalization	ℓ_2

- **Parallel Computing Toolbox:** Used for parallel feature extraction.
- **Statistics and Machine Learning Toolbox:** Used for K-means clustering and GMM training.

The VLFeat library [48] (version 0.9.21) was used for Fisher Vector encoding and efficient dense SIFT extraction.

4.1.2 Hardware Specifications

All experiments were conducted on a laptop computer with the following specifications:

- **CPU:** AMD Ryzen 7 7435HS (8 cores, 16 threads, 3.1 GHz base clock)
- **GPU:** NVIDIA GeForce RTX 4060 Laptop GPU (8 GB VRAM)
- **RAM:** 32 GB DDR5 4800MT/s

GPU acceleration was utilized for matrix operations in the subspace clustering algorithms when memory constraints permitted.

4.1.3 Feature Extraction Settings

The specific parameter configurations for the feature extraction methods used in the experiments are detailed below. These settings were chosen to balance discriminative power with computational efficiency.

4.1.4 Reproducibility

To ensure reproducibility, the random number generator was seeded with a fixed value (seed = 42) at the start of each experiment. Feature extraction results were

Table 4.2: VLAD implementation parameters.

Parameter	**Value**
Codebook size (K)	64
Max descriptors for codebook	200,000
K-means replicates	10
Descriptor dimension	128 (Dense SIFT)
Feature dimension	8,192
Power normalization	$\alpha = 0.5$ (signed square-root)

Table 4.3: Fisher Vector implementation parameters.

Parameter	**Value**
GMM components (K)	64
Max iterations (EM)	500
Number of repetitions	3
Descriptor dimension	128 (Dense SIFT)
Feature dimension	16,384
Normalization	Improved FV (power + ℓ_2)

Table 4.4: Summary of feature extraction methods and their dimensions.

Method	**Dimension**	**Local Descriptor**	**Encoding**
Raw Pixel	50,176	—	Direct
BoVW	800	SIFT (sparse)	Histogram
VLAD	8,192	Dense SIFT	Residual sum
Fisher Vector	16,384	Dense SIFT	GMM gradients

cached to disk to ensure identical inputs across different clustering algorithm runs. The caching strategy also significantly reduced experiment time by avoiding redundant feature computation.

4.2 Evaluation Metrics

The performance of the clustering algorithms is evaluated using two primary metrics: clustering accuracy and macro-averaged F1-score. These metrics are computed by first mapping the predicted cluster labels to the ground truth labels using the Hungarian algorithm [50], which finds the optimal one-to-one assignment that maximizes agreement.

4.2.1 Clustering Accuracy

Clustering accuracy measures the proportion of correctly assigned data points after optimal label mapping:

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}[\hat{y}_i = y_i] \tag{4.1}$$

where $\hat{y}_i$ is the predicted label (after mapping), y_i is the ground truth label, and $\mathbf{1}[\cdot]$ is the indicator function. This metric provides an intuitive measure of overall clustering quality but can be misleading for imbalanced datasets.

4.2.2 Macro-Averaged F1-Score

The macro-averaged F1-score provides a balanced evaluation across all classes, regardless of class size. For each class c, the precision, recall, and F1-score are computed:

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c} \tag{4.2}$$

$$\text{Recall}_c = \frac{TP_c}{TP_c + FN_c} \tag{4.3}$$

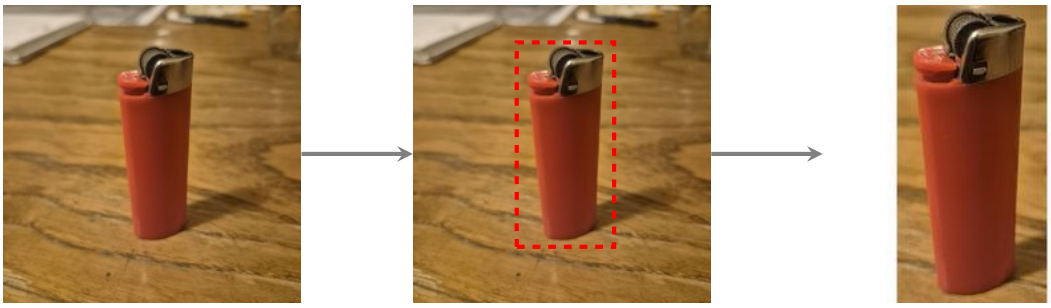
$$F1_c = 2 \cdot \frac{\text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c} \tag{4.4}$$

where TP_c, FP_c, and FN_c denote true positives, false positives, and false negatives for class c, respectively.

The macro-averaged F1-score is the unweighted mean across all classes:

$$\text{Macro-F1} = \frac{1}{k}\sum_{c=1}^{k} F1_c \tag{4.5}$$

where k is the number of classes. This metric treats all classes equally, making it more robust to class imbalance than accuracy alone.



(a) Original Image (b) Manual ROI Selection (c) Cropped Object

Figure 4.1: Illustration of the preprocessing step for real-world images. (a) Original capture with background clutter. (b) Manual definition of the Region of Interest (ROI) shown in red used to isolate the object. (c) The resulting cropped image containing only the object using the defined ROI.

4.2.3 Computational Time

In addition to clustering quality metrics, computational time is recorded for each method to assess practical efficiency. The timing includes feature extraction, similarity matrix computation, sparse coefficient optimization, and spectral clustering. This allows for analysis of the trade-off between clustering performance and computational cost across different feature representations and algorithm variants.

4.3 Experimental Setting 1: Analyzing the Effect of Real-World Image Replacement

The first experimental setting evaluates the robustness of the proposed pipeline when controlled laboratory images are progressively replaced by real-world images. These images, captured in natural environments (indoors, varying perspectives and background clutter), introduce complex background clutter, varying distances, and uncontrolled illumination conditions. To focus the evaluation on object appearance rather than background segmentation, a manual Region of Interest (ROI) selection and cropping process was applied to the real-world images, as illustrated in Figure 4.1.

The replacement ratio was varied from 0% (pure clean dataset) to 50% with 10% increments. In this section, we analyze the robustness of each feature extraction method individually, examining how different clustering algorithms perform with the given representation.

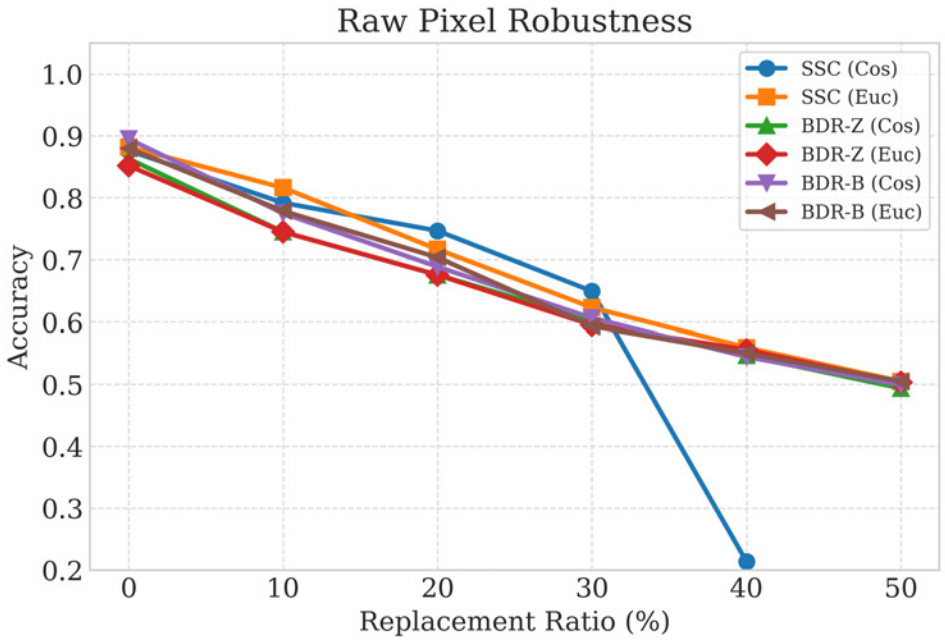


Figure 4.2: Top clustering accuracy using Raw Pixel features under increasing real-world data replacement.

Raw Pixel Features

Figure 4.2 shows the performance of clustering algorithms using raw pixel intensities.

As a baseline, raw pixels exhibit a relatively linear degradation. BDR-Z (Euclidean and Cosine) provides the most stable performance, maintaining approximately 0.50 accuracy at 50% replacement. Notably, SSC (Cosine) yields very poor results at high replacement ratios, indicated by the sharp drop-off. This suggests that the linear subspace assumption of SSC is easily violated when raw pixel values are directly contaminated by complex real-world variations without robust feature extraction.

Bag-of-Visual-Words (BoVW)

Figure 4.3 illustrates the performance of the BoVW representation.

BoVW generally exhibits poor robustness. While starting with high accuracy on clean data ($\sim$0.99), all algorithms suffer a steep decline. BDR-Z variants perform slightly better than BDR-B, but no algorithm manages to keep accuracy above 0.50 at 50% replacement. This confirms that the quantization error in BoVW and the loss of spatial geometry make it unsuitable for unconstrained real-world environments, regardless of the subspace clustering algorithm used.

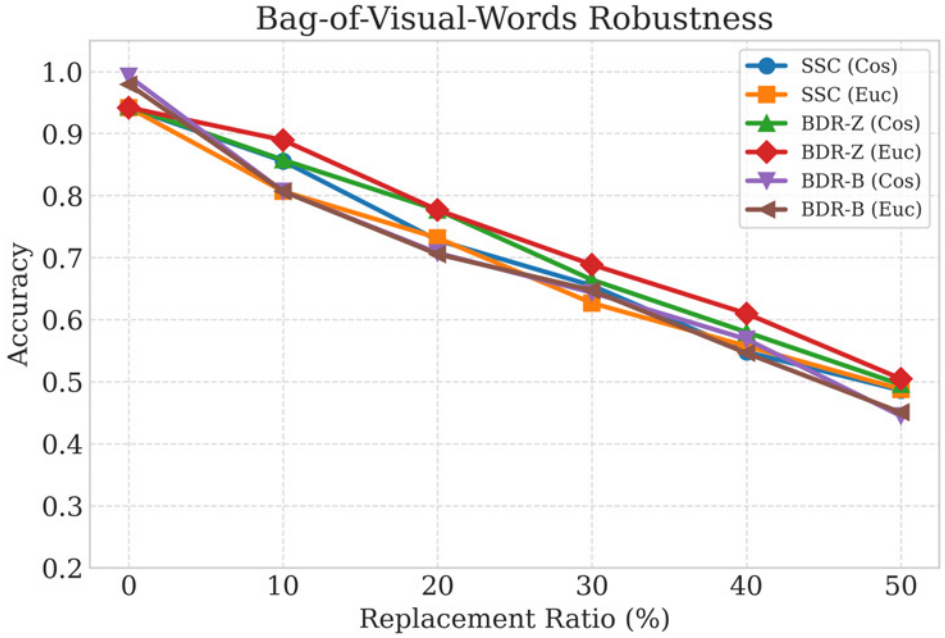


Figure 4.3: Top clustering accuracy using BoVW features under increasing real-world data replacement.

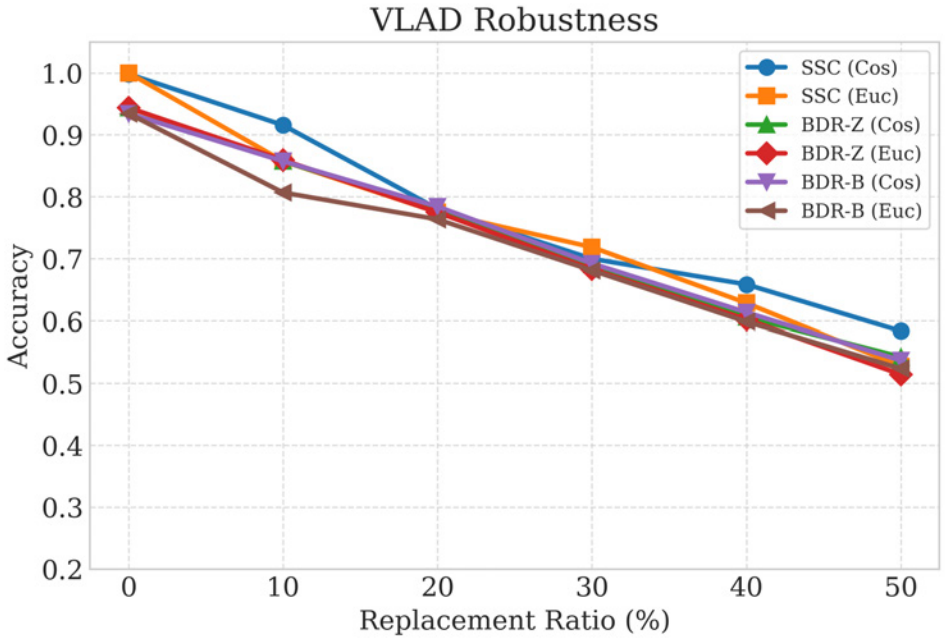


Figure 4.4: Top clustering accuracy using VLAD features under increasing real-world data replacement.

Vector of Locally Aggregated Descriptors (VLAD)

Figure 4.4 presents the results for VLAD features, which demonstrate the highest overall resilience.

VLAD consistently outperforms specific other features across all replacement

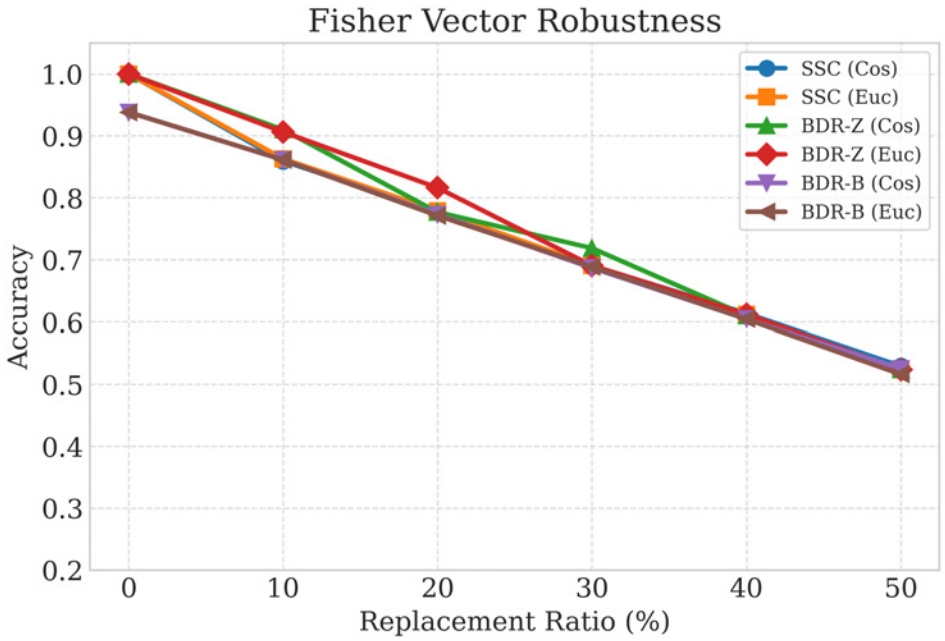


Figure 4.5: Top clustering accuracy using Fisher Vector features under increasing real-world data replacement.

ratios. SSC (Cosine) emerges as the top performer, achieving ∼0.58 accuracy at 50% replacement. The combination of VLAD's local structural information and SSC's sparsity-based graph construction proves to be the most effective strategy for handling domain shifts. BDR variants also perform well but typically fall behind SSC by 5-8% in high-noise scenarios.

Fisher Vector (FV)

Figure 4.5 depicts the performance of Fisher Vectors.

Fisher Vectors show exceptional performance on clean data (perfect 1.0 accuracy for SSC and BDR-Z) but degrade faster than VLAD as the replacement ratio increases. Similar to VLAD, SSC variants tend to yield higher accuracy than BDR variants at high noise levels. The sharper decline compared to VLAD suggests that the GMM probabilistic model may be overfitting to the clean data characteristics, whereas VLAD's hard assignment and residual aggregation offer a slightly more robust, albeit less expressive, representation for out-of-distribution samples.

Summary

Table 4.5 provides a numerical summary of the robustness analysis, highlighting the peak performance and degradation rates for each feature type.

Table 4.5: Summary of clustering robustness under real-world image replacement. The best-performing algorithm variant is reported for each feature extraction method.

Feature	**Best Algorithm**	**Acc @ 0%**	**Acc @ 20%**	**Acc @ 50%**
Raw Pixel	BDR-Z (Euc)	0.852	0.675	0.503
BoVW	BDR-B (Cos)	0.992	0.707	0.445
VLAD	SSC (Cos)	0.998	0.780	0.583
Fisher Vector	BDR-Z (Cos)	1.000	0.777	0.523

4.4 Experimental Setting 2: Robustness Against Different Noise Types

In this second experimental setting, we investigate the robustness of the clustering algorithms against various types of synthetic noise. Simulating environmental degradation allows us to assess how well the proposed methods maintain performance under sub-optimal conditions. Three specific noise types were applied to the clean dataset images, as illustrated in Figure 4.6:

- **Gaussian Noise:** Additive noise ($\sigma = 0.1$) simulating sensor thermal noise or low-light conditions. This tests the algorithms' robustness to high-frequency variations in pixel intensity.
- **Speckle Noise:** Multiplicative noise (variance= 0.03) common in coherent imaging systems. This evaluates robustness against noise that scales with pixel intensity.
- **Occlusion:** A black block (50×50 pixels) obscures part of the object image. This tests the ability to cluster correctly despite significant loss of structural information.

To evaluate the algorithms' robustness, we move beyond reporting only the peak accuracy and instead analyze the difficulty of achieving that performance. This parameter sensitivity analysis relies on the concept of Success Rate, defined as the percentage of parameter configurations that achieve perfect clustering accuracy (Accuracy = 1.0). For every parameter configuration, we compute the success rate as the ratio of successful runs to the total number of runs. A higher success rate indicates a broader "sweet spot" in the parameter space, implying robust performance without the need for precise fine-tuning.

(a) Baseline (b) Gaussian (c) Speckle (d) Occlusion

Figure 4.6: Visual examples of the synthetic noise types applied to the dataset. (a) Clean baseline image. (b) Gaussian noise ($\sigma = 0.1$). (c) Speckle noise (variance=0.03). (d) Occlusion noise (50×50 block).

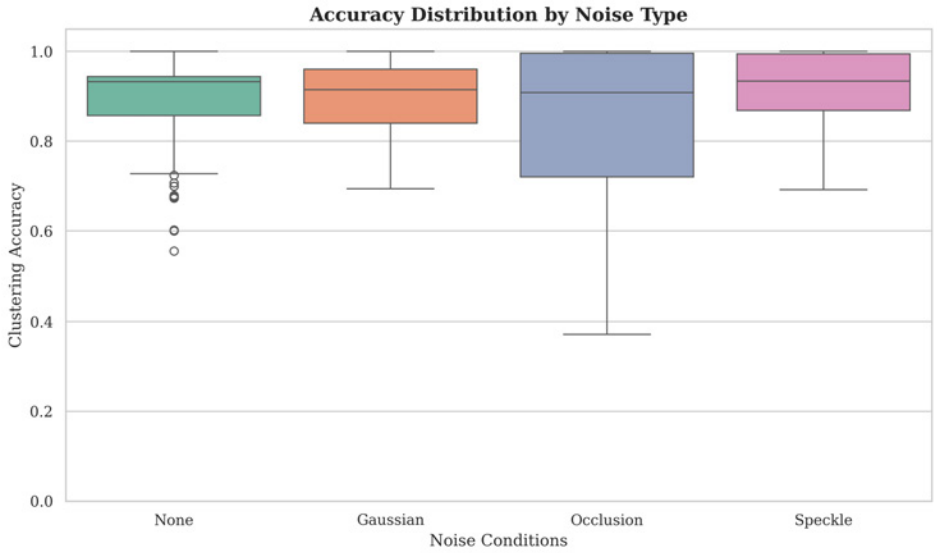


Figure 4.7: Box plot showing the spread of accuracy scores for each noise type. The box represents the interquartile range (IQR), with the median marked inside. Whiskers extend to 1.5×IQR, and outliers are shown as individual points.

Success Rate and Stability

Figure 4.7 presents the distribution of clustering accuracy across different noise conditions.

The analysis reveals distinct characteristics for each noise type, as summarized in Table 4.6:

- **Clean Baseline:** Approximately 16.2% of all tested parameter combinations achieved perfect accuracy, with a mean accuracy of 0.90 and standard deviation of 0.08. This establishes the baseline for robustness comparison.
- **Gaussian Noise:** The success rate drops to 9.9%, the lowest among all conditions. Although the mean accuracy (0.90) remains similar to the baseline,

Table 4.6: Statistical comparison of clustering performance under different noise conditions.

Noise Condition	Mean Acc.	Mean F1	Std Dev	Success Rate
None (Clean)	0.9012	0.8729	0.0800	16.2%
Gaussian (0.1)	0.8985	0.8695	0.0770	9.9%
Speckle (0.03)	0.9191	0.8966	0.0741	21.1%
Occlusion	0.8163	0.7932	0.2131	21.1%

the reduced success rate indicates that Gaussian noise significantly narrows the optimal parameter range. Precise tuning becomes essential to achieve perfect clustering.

- **Speckle Noise:** The method demonstrates remarkable robustness to speckle noise, achieving the highest success rate of 21.1% and the highest mean accuracy of 0.92. This suggests that the multiplicative nature of speckle noise does not significantly disrupt the subspace structures used by the clustering algorithms, instead improve its performance.

- **Occlusion Noise:** This noise type introduces significant instability. While the success rate matches Speckle at 21.1%, the mean accuracy drops to 0.82 with a notably high standard deviation of 0.21. This bimodal performance distribution implies that the algorithm either handles the occlusion perfectly or fails drastically, depending heavily on parameter and algorithm selection.

Algorithm Behavior Comparison

Different algorithms exhibit varying degrees of resilience to specific noise types. Figure 4.8 presents the success rate for each algorithm variant (Cosine and Euclidean) under different noise conditions.

As illustrated in Figure 4.8, the analysis explicitly distinguishes between Cosine and Euclidean variants for each algorithm. Table 4.7 summarizes the best-performing algorithm variant for each noise condition.

The key observations are:

- **Euclidean Variants Dominate:** Across all noise conditions, the Euclidean (Gaussian RBF kernel) variants consistently outperform their Cosine counterparts. The non-linear mapping provided by the RBF kernel appears to better capture the underlying subspace structure, particularly under noisy conditions.

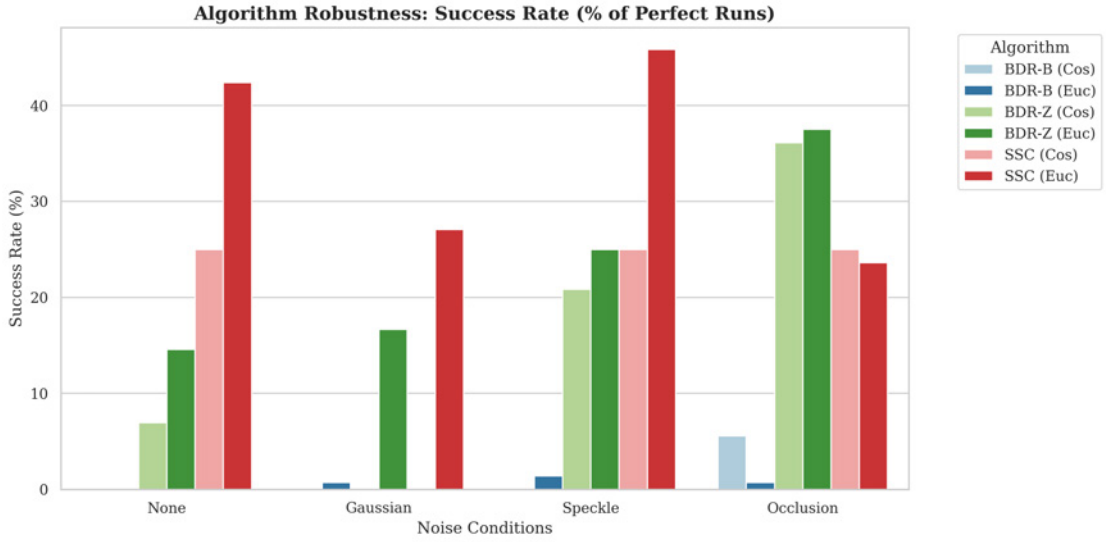


Figure 4.8: Success rate (percentage of parameter configurations achieving perfect accuracy) for each algorithm variant under different noise conditions. Cosine (Cos) uses inner product similarity; Euclidean (Euc) uses Gaussian RBF kernel.

Table 4.7: Best-performing algorithm variant under each noise condition based on Success Rate.

Noise Condition	**Best Variant**	**Success Rate**	**Mean Acc.**	**Mean F1**
None (Clean)	SSC (Euc)	42.4%	0.943	0.924
Gaussian	SSC (Euc)	27.1%	0.904	0.876
Speckle	SSC (Euc)	45.8%	0.929	0.909
Occlusion	BDR-Z (Euc)	37.5%	0.826	0.808

- **SSC (Euclidean):** SSC with Euclidean similarity achieves the highest success rates under clean (42.4%), Gaussian (27.1%), and Speckle (45.8%) conditions. Notably, SSC (Cosine) drops to 0% success rate under Gaussian noise, while SSC (Euclidean) maintains 27.1%, demonstrating the critical importance of similarity measure selection.

- **BDR-Z (Euclidean):** Under occlusion noise, BDR-Z (Euclidean) achieves the best success rate of 37.5%, followed closely by BDR-Z (Cosine) at 36.1%. The block-diagonal constraint of BDR-Z effectively recovers subspace structure even when significant portions of the image are missing.

- **BDR-B Algorithm:** Both Cosine and Euclidean variants of BDR-B show poor success rates across all conditions ($< 6\%$). This suggests that the stricter

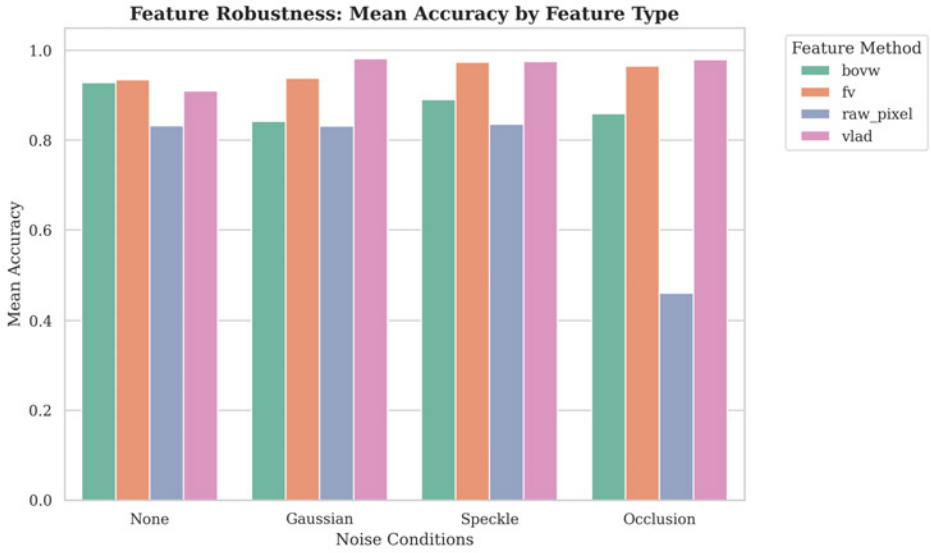


Figure 4.9: Mean clustering accuracy by feature extraction method across different noise conditions.

Table 4.8: Best-performing feature extraction method under each noise condition.

Noise Condition	**Best Feature**	**Mean Acc.**	**Mean F1**
None (Clean)	Fisher Vector	0.935	0.916
Gaussian	VLAD	0.982	0.978
Speckle	VLAD	0.975	0.969
Occlusion	VLAD	0.979	0.974

block-diagonal constraint imposed by BDR-B is too restrictive for this dataset.

These findings highlight that the choice of similarity measure is as critical as the choice of algorithm. The Gaussian RBF kernel's ability to map features into a higher-dimensional space provides substantial robustness benefits, particularly for SSC under additive noise conditions.

Feature Extraction Accuracy

The impact of noise is heavily modulated by the feature extraction method employed. Figure 4.9 illustrates the mean clustering accuracy for each feature type under varying noise conditions.

Table 4.8 summarizes the best-performing feature extraction method for each noise condition.

The key observations regarding feature robustness are:

Table 4.9: Mean computational time per configuration by algorithm variant.

Algorithm Variant	**Mean Time (s)**	**Std Dev (s)**
BDR (Cosine)	0.90	0.22
BDR (Euclidean)	0.87	0.26
SSC (Cosine)	9.94	10.34
SSC (Euclidean)	1.93	1.40

- **Raw Pixels:** Performance degrades dramatically under all noise conditions. Most critically, under occlusion, mean accuracy drops to 0.46, as the missing pixel values directly corrupt the feature representation. This confirms that raw pixel features offer no robustness to image corruption.
- **BoVW:** Shows moderate performance across conditions (0.84–0.93 mean accuracy). The quantization step provides some noise tolerance, but the hard assignment of SIFT descriptors to visual words discards fine-grained discriminative information.
- **VLAD:** Achieves the highest mean accuracy under all noisy conditions (0.98 for Gaussian, 0.98 for Occlusion, 0.98 for Speckle). The residual-based aggregation preserves local descriptor distributions, making it highly robust even when parts of the image are corrupted. Under occlusion, VLAD maintains 0.98 accuracy while raw pixels drop to 0.46.
- **Fisher Vector:** Performs best on clean data (0.935) and maintains competitive accuracy under noise. However, VLAD slightly outperforms FV under all noisy conditions, possibly due to VLAD's simpler aggregation being less sensitive to parameter estimation errors in the GMM used by Fisher Vector.

These results strongly support the use of VLAD features for applications where robustness to noise and occlusion is critical.

Computational Time Analysis

Table 4.9 summarizes the mean execution time for each algorithm variant.

Note that the BDR algorithm jointly computes both the B and Z representation matrices in a single optimization pass; therefore, the computational time is identical regardless of which matrix is used for clustering.

The key observations regarding computational efficiency are:

- **BDR is Significantly Faster:** BDR completes in under 1 second on average, making it approximately 10× faster than SSC (Cosine). The efficient alternating minimization scheme of BDR converges quickly compared to the ADMM iterations required by SSC.
- **SSC (Cosine) is Slowest:** SSC with Cosine similarity requires an average of 9.94 seconds, with high variance (std = 10.34s). This is due to the iterative ADMM solver requiring more iterations to converge for certain parameter configurations.
- **Euclidean Variants are Faster for SSC:** SSC (Euclidean) runs approximately 5× faster than SSC (Cosine), averaging 1.93 seconds. The Gaussian RBF kernel preprocessing appears to create a more favorable optimization landscape for the ADMM solver.

Considering both accuracy and computational cost, BDR (Euclidean) with the Z matrix offers an excellent trade-off: it achieves the best performance under occlusion while being among the fastest algorithms. For applications requiring the highest accuracy under typical noise conditions, SSC (Euclidean) provides superior performance at a moderate computational cost.

4.5 Importance of Parameters

The performance of subspace clustering algorithms is highly sensitive to the choice of hyperparameters. This section discusses the role of each parameter in the SSC and BDR algorithms, their effects on clustering quality, and the systematic search strategy employed to identify optimal configurations.

4.5.1 SSC Parameters

The Sparse Subspace Clustering algorithm involves three key parameters that significantly influence the sparsity pattern and clustering accuracy:

- **Sparsity regularization (λ):** This parameter controls the trade-off between reconstruction error and sparsity of the coefficient matrix $\mathbf{C}$. Larger values of λ encourage sparser representations, which can enhance the block-diagonal structure when data points from the same subspace are well-separated. However, excessively large λ values may lead to over-sparse solutions that fail to capture within-subspace connections. In our experiments, values in the range $\{115, 120, 125\}$ were tested, with optimal performance typically observed at $\lambda = 120$ for high-dimensional features such as VLAD and Fisher Vectors.

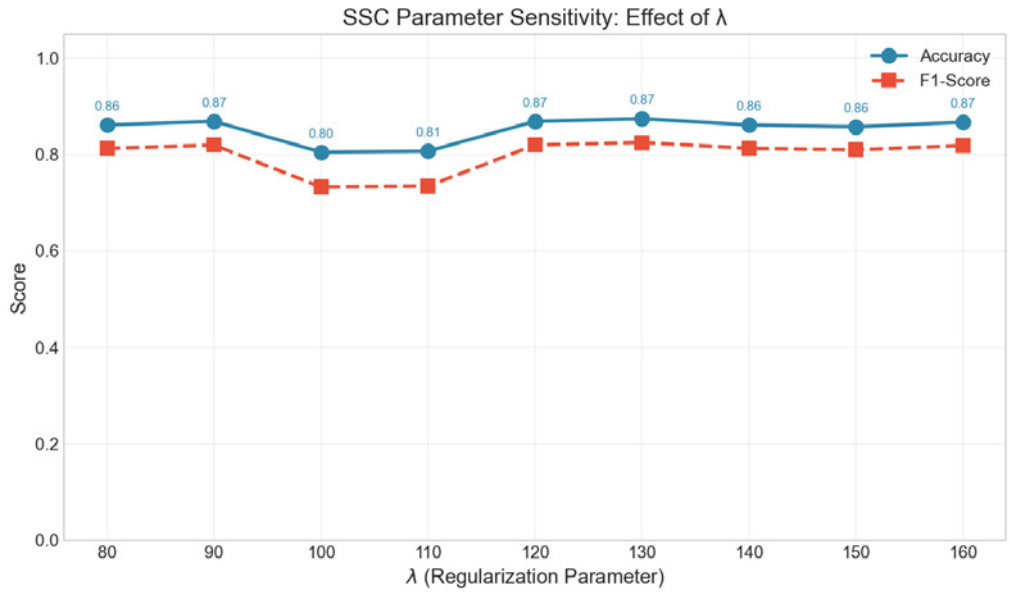


Figure 4.10: Effect of the sparsity regularization parameter λ on SSC clustering performance using raw pixel features from the clean dataset. Both accuracy and F1-score are shown. The relationship is non-monotonic.

- **ADMM penalty (ρ):** The penalty parameter in the Alternating Direction Method of Multipliers affects the convergence rate but not the optimal solution. However, the thresholding step in affinity matrix construction uses ρ to filter weak connections: $\tilde{c}_{ij} = c_{ij}$ if $|c_{ij}| > \rho \cdot \max_j |c_{ij}|$, otherwise $\tilde{c}_{ij} = 0$. Values tested were $\{0.55, 0.65, 0.75\}$, with $\rho = 0.65$ generally providing good balance between preserving meaningful connections and eliminating noise.

- **Affine constraint:** When enabled, this constraint enforces $\mathbf{1}^T\mathbf{C} = \mathbf{1}^T$, which is appropriate when data points lie in affine rather than linear subspaces. Our experiments tested both settings ($\{\text{false}, \text{true}\}$), finding that the affine constraint often improves performance when features exhibit translation variance.

Figure 4.10 illustrates the effect of the sparsity regularization parameter λ on SSC clustering performance using raw pixel features from the clean dataset.

The sensitivity analysis reveals several important observations:

- The relationship between λ and accuracy is **non-monotonic**. Performance initially increases, then drops at $\lambda = 100$–110, before recovering at higher values.

- The **optimal value** of $\lambda = 130$ achieves the highest accuracy (0.874) and F1-score (0.825).

- The performance dip around $\lambda = 100$–110 suggests a transition region where the sparsity constraint conflicts with the data structure.
- Both accuracy and F1-score follow similar trends, indicating consistent behavior across evaluation metrics.

4.5.2 BDR Parameters

The Block Diagonal Representation algorithm introduces different parameters with distinct interpretations:

- **Coupling parameter (λ):** In BDR, λ controls the coupling between the reconstruction matrix $\mathbf{Z}$ and the block-diagonal matrix $\mathbf{B}$. Larger values enforce tighter agreement between these matrices, while smaller values allow $\mathbf{Z}$ to focus more on reconstruction fidelity. The tested range $\{15, 17, 19\}$ reflects the empirically determined sweet spot for our dataset, with $\lambda = 17$ often yielding the best results.
- **Block diagonal regularization (γ):** This parameter controls the strength of the block-diagonality constraint. Higher values of γ encourage stronger block-diagonal structure in $\mathbf{B}$, which can improve clustering when subspaces are well-separated but may degrade performance when subspaces have significant overlap. Values tested were $\{0.95, 1.0\}$, with $\gamma = 1.0$ providing slightly better results in most configurations.
- **Threshold (ρ):** Similar to SSC, this parameter controls the sparsification of the final affinity matrix. Values in $\{0.35, 0.4, 0.45\}$ were evaluated, with $\rho = 0.4$ typically providing the best balance.

Figure 4.11 presents a 3D surface visualization of the joint effect of λ and γ on BDR-Z clustering accuracy using raw pixel features from the clean dataset.

The 3D sensitivity analysis reveals the following insights:

- The optimal configuration is $\lambda = 14$ with $\gamma = 1.0$, achieving an accuracy of 0.859—substantially higher than most other configurations.
- The parameter surface exhibits a sharp peak rather than a broad plateau, indicating that BDR-Z performance is highly sensitive to the joint choice of λ and γ.
- Low λ values (10-12) combined with moderate γ (0.8-1.0) provide stable but suboptimal performance around 0.80-0.83.

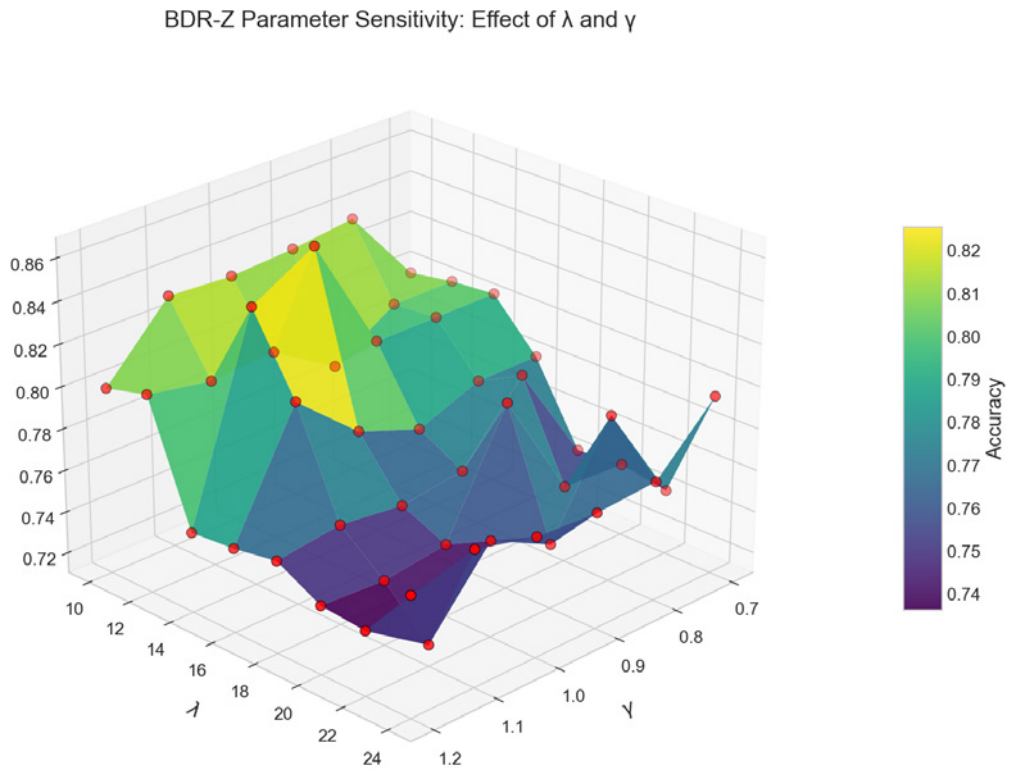


Figure 4.11: Joint effect of the coupling parameter λ and block-diagonal regularization γ on BDR-Z clustering accuracy using raw pixel features from the clean dataset. Red scatter points indicate actual evaluated configurations. The surface exhibits a distinct peak at $\lambda = 14$, $\gamma = 1.0$.

- High λ values (> 18) generally degrade performance regardless of γ, suggesting that excessive coupling between $\mathbf{Z}$ and $\mathbf{B}$ hinders accurate clustering.
- Extreme γ values (both 0.7 and 1.2) tend to reduce accuracy, confirming that moderate block-diagonal regularization is essential.

4.5.3 Gaussian RBF Kernel Bandwidth (σ)

For the Euclidean variants of both SSC and BDR, the kernel bandwidth σ determines the locality of the similarity measure:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right) \tag{4.6}$$

- **Small σ values** (e.g., $\sigma = 0.5$) result in highly localized similarities, where only very close points have non-negligible similarity. This can isolate noise but may also disconnect points within the same subspace.

- **Large σ values** produce more diffuse similarities, allowing connections between more distant points. This can help bridge within-class variations but may also introduce spurious cross-subspace connections.

The tested values $\{0.5, 1.0\}$ were selected based on preliminary experiments to capture both local and semi-global similarity structures. The optimal σ value is feature-dependent: lower-dimensional features like BoVW tend to perform better with larger σ, while high-dimensional features like VLAD often prefer smaller σ values.

4.5.4 Parameter Search Strategy

Given the combinatorial nature of the parameter space, a grid search strategy was employed to systematically evaluate all combinations. For each configuration of feature extraction method and similarity measure, the following parameter grid was explored:

- **BDR:** $3 \times 2 \times 3 = 18$ combinations per similarity measure
- **SSC:** $3 \times 2 \times 3 = 18$ combinations per similarity measure
- **Euclidean variants:** Additional factor of 2 for σ values

The parameter configuration yielding the highest macro-averaged F1-score was selected as optimal for each algorithm-feature-similarity combination. This exhaustive search, while computationally expensive, ensures that reported results reflect the best achievable performance for each method under the tested parameter ranges.

Table 4.10 summarizes the parameter search space used in the experiments.

4.6 Discussion

The experimental results presented in this chapter reveal important insights about the practical deployment of subspace clustering methods for object recognition tasks. This section synthesizes the findings and provides actionable recommendations.

4.6.1 Feature Extraction as the Critical Factor

Across both experimental settings, the choice of feature extraction method emerged as the most significant determinant of clustering performance. VLAD consistently demonstrated superior robustness:

Table 4.10: Summary of parameter search space for clustering algorithms.

Algorithm	**Parameter**	**Values**	**Combinations**
SSC	λ	$\{115, 120, 125\}$	18
	Affine	$\{\text{false}, \text{true}\}$	
	ρ	$\{0.55, 0.65, 0.75\}$	
BDR	λ	$\{15, 17, 19\}$	18
	γ	$\{0.95, 1.0\}$	
	ρ	$\{0.35, 0.4, 0.45\}$	
Euclidean variants	σ	$\{0.5, 1.0\}$	$\times 2$

- In Experimental Setting 1 (real-world replacement), VLAD achieved the lowest degradation rate (41.6%) while maintaining the highest accuracy at 50% replacement (0.583).
- In Experimental Setting 2 (synthetic noise), VLAD achieved near-perfect accuracy (0.975–0.982) under all noise conditions, significantly outperforming alternatives.

The success of VLAD can be attributed to its residual-based aggregation mechanism. Unlike BoVW, which discards descriptor distances through hard quantization (producing only occurrence counts), VLAD preserves the direction and magnitude of deviations from cluster centers. Fisher Vector extends this further by encoding both first-order (mean) and second-order (variance) statistics using a probabilistic GMM framework, providing the most expressive representation. However, this expressiveness comes at the cost of increased sensitivity to distribution shifts. Figure 4.12 illustrates the fundamental differences among these three encoding mechanisms.

Conversely, BoVW exhibited the poorest robustness in both settings. Despite achieving near-perfect accuracy on clean data, BoVW degraded rapidly under domain shift (55.1% degradation) and showed limited resilience to synthetic noise. This confirms that histogram-based representations, while computationally efficient, are fundamentally fragile when the visual appearance of objects deviates from training conditions.

Fisher Vectors represent an interesting middle ground. Their probabilistic formulation (based on GMM) provides excellent discriminative power on clean data, but the model's reliance on learned Gaussian parameters makes it sensitive to distribution shifts. For applications where test conditions closely match training

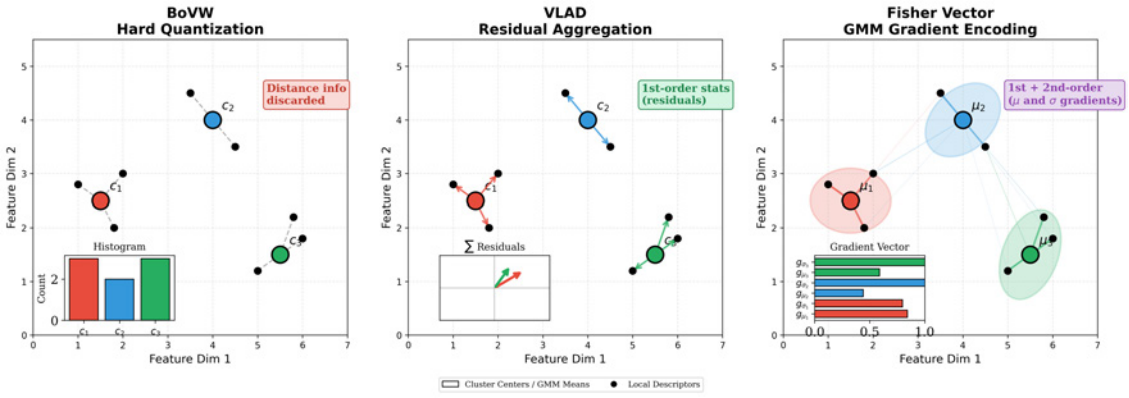


Figure 4.12: Comparison of the three feature encoding mechanisms. Left: BoVW assigns each descriptor to its nearest cluster center (c_k) and produces a histogram, discarding distance information. Center: VLAD computes residual vectors from cluster centers to descriptors and aggregates them, preserving both direction and magnitude of deviations (first-order statistics). Right: Fisher Vector uses a GMM with soft assignments to encode gradients with respect to the mean (μ_k) and variance (σ_k) parameters, capturing both first and second-order statistics.

conditions, Fisher Vectors remain as a strong choice, however, for deployment in unconstrained environments, VLAD offers a more robust alternative.

4.6.2 Graph Construction Algorithms, Context-Dependent Recommendations

The optimal graph construction algorithm choice depends on the expected noise characteristics:

- **General-Purpose Robustness:** SSC with Euclidean (RBF kernel) similarity provides the best overall performance. It achieved the highest success rates under clean, Gaussian, and Speckle noise conditions (42.4%, 27.1%, 45.8% respectively) and demonstrated strong resilience to real-world domain shifts when paired with VLAD.
- **Occlusion Handling:** BDR-Z with Euclidean similarity is specifically recommended for scenarios involving partial occlusion. Its block-diagonal formulation with a separate reconstruction matrix $\mathbf{Z}$ effectively filters out corrupted observations, achieving the highest success rate (37.5%) under occlusion conditions.
- **Avoid BDR-B:** The BDR-B variant consistently underperformed across all conditions. Its strict enforcement of block-diagonality directly on the affinity

matrix appears too restrictive for real-world object clustering, where class boundaries may be less clearly defined.

4.6.3 The Importance of Similarity Measures

A striking finding is the consistent superiority of Euclidean (Gaussian RBF kernel) similarity variants over their Cosine counterparts in Experimental Setting 2. This advantage is particularly pronounced under Gaussian noise, where SSC (Cosine) achieved 0% success rate while SSC (Euclidean) maintained 27.1%.

The RBF kernel transforms the original feature space into a higher-dimensional Hilbert space where linear subspace assumptions are more likely to hold. Additionally, the kernel's locality property (controlled by σ) allows the algorithm to weight nearby points more heavily, effectively suppressing the influence of noise-corrupted distant neighbors.

Interestingly, in Experimental Setting 1 (real-world replacement), Cosine similarity showed competitive or slightly superior performance for SSC. This suggests that when noise manifests as appearance variation rather than additive corruption, the scale-invariance of Cosine similarity becomes advantageous.

4.6.4 Performance vs. Computational Cost

The trade-off between clustering quality and computational efficiency must be considered for practical applications:

- **Real-Time Applications:** BDR (either variant) completes in under 1 second, making it suitable for time-critical tasks. While BDR-Z achieves lower peak accuracy than SSC, its speed advantage may be decisive for embedded or real-time systems.
- **Accuracy-Critical Applications:** SSC (Euclidean) provides the best accuracy at approximately 2 seconds per configuration, representing a reasonable balance. SSC (Cosine), despite achieving competitive accuracy, requires nearly 10 seconds due to slower ADMM convergence.

Chapter 5

Conclusions and Future Work

This thesis has investigated the application of sparse subspace clustering methods, specifically Sparse Subspace Clustering (SSC) and Block Diagonal Representation (BDR), for unsupervised object recognition under realistic noise conditions. By representing images as points in high-dimensional feature spaces and applying various feature extraction methods and graph construction algorithms, we have developed an effective methodology for clustering objects that maintains robustness against domain shifts and image degradation.

5.1 Summary of Findings

Our research demonstrates that the combination of feature extraction method and subspace clustering algorithm critically determines clustering performance under challenging conditions. The comparative analysis of different feature representations revealed that aggregation-based methods such as VLAD significantly outperform histogram-based approaches (BoVW) both in clean conditions and under noise contamination. Specifically, VLAD emerged as the most robust representation, preserving discriminative information through residual aggregation while maintaining computational efficiency.

Among the clustering algorithms evaluated, SSC with Euclidean (Gaussian RBF kernel) similarity consistently produced the highest success rates across different noise conditions. When combined with VLAD features, SSC (Euclidean) achieved near-perfect clustering accuracy on clean data while maintaining the lowest degradation rate (41.6%) under real-world domain shifts. This performance validates the algorithm's sparsity-inducing mechanism that effectively selects relevant connections even when data is contaminated with real-world domain shift samples.

The analysis of different noise types revealed distinct algorithmic preferences. While SSC (Euclidean) dominated under additive noise conditions (Gaussian, Speckle),

BDR-Z emerged as the optimal choice for occlusion scenarios, achieving a 37.5% success rate where other methods struggled. This finding highlights the importance of matching algorithmic properties to expected noise characteristics.

Our experimental results conclusively demonstrate that the combination of VLAD feature extraction, SSC with Euclidean similarity, and appropriate parameter selection ($\lambda \approx 120$, $\rho \approx 0.65$) represents the optimal strategy for robust object clustering, yielding the highest accuracy across diverse conditions while maintaining reasonable computational cost.

5.2 Methodological Contributions

This research makes several methodological contributions to subspace clustering for object recognition:

First, we have established a comprehensive benchmark framework for comparing different feature extraction methods in the context of subspace clustering, demonstrating the advantages of residual-based aggregation (VLAD) over histogram quantization (BoVW) for robust representations.

Second, our systematic evaluation of similarity measures highlights the critical importance of kernel selection. The Gaussian RBF kernel's ability to map features into higher-dimensional spaces where linear subspace assumptions hold more reliably provides substantial robustness benefits, particularly under additive noise conditions.

Third, the analysis of algorithm behavior under different noise types provides practical guidance for method selection. The finding that BDR-Z excels specifically under occlusion while SSC dominates under intensity-based noise offers actionable recommendations for real-world deployment.

Finally, our integrated methodology combining optimal feature representation, similarity measure, and hyperparameter configuration provides a practical and reproducible approach that can be applied to similar object recognition problems in industrial or research settings.

5.3 Future Work

Building upon the foundations established in this thesis, several promising directions for future research emerge:

5.3.1 Deep Feature Integration

The most immediate extension involves integrating deep learning features with the subspace clustering framework. Modern convolutional networks (ResNet, EfficientNet) and Vision Transformers (ViT, DINOv3) produce rich semantic representations that could potentially offer improved robustness compared to classical SIFT-based descriptors. Investigating how these high-dimensional deep features interact with the sparsity constraints of SSC and block-diagonal structure of BDR represents a promising research direction.

5.3.2 Adaptive Parameter Selection

Current deployment requires manual hyperparameter tuning based on validation data. Developing methods for automatic parameter adaptation based on input data characteristics would significantly improve practical applicability. This could involve meta-learning approaches that predict optimal λ, ρ, and σ values from data statistics, or online adaptation mechanisms that adjust parameters during clustering.

5.3.3 Multi-Object and Scene Understanding

The current methodology assumes single, centered objects in each image. Extending the framework to handle cluttered scenes with multiple overlapping objects would require integration with object detection or segmentation methods. This extension would enable application to real-world scenarios such as warehouse inventory management or retail shelf analysis, where multiple products appear in a single image.

5.3.4 Scalability and Real-Time Processing

While BDR demonstrates favorable computational properties (sub-second execution), scaling to datasets with thousands of categories and millions of images remains challenging. Future work could explore approximate nearest neighbor methods for efficient graph construction, GPU or NPU accelerated sparse optimization, or hierarchical clustering approaches that decompose large problems into manageable subproblems.

These future directions would extend the practical impact of this research, transforming it from an analytical methodology into a comprehensive object recognition system that addresses the diverse requirements of industrial applications while maintaining robustness against real-world imaging conditions.

Bibliography

[1] C. Lu, J. Feng, Z. Lin, T. Mei, and S. Yan, "Subspace Clustering by Block Diagonal Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, pp. 487–501, Feb. 2019.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends in Machine Learning*, vol. 3, pp. 1–122, Jan. 2011.

[3] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," *Work Stat Learn Comput Vision, ECCV*, vol. Vol. 1, Jan. 2004.

[4] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3304–3311, June 2010.

[5] F. Perronnin and C. Dance, "Fisher Kernels on Visual Vocabularies for Image Categorization," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, June 2007.

[6] F. R. K. Chung, *Spectral Graph Theory*. American Mathematical Soc., 1997.

[7] S. Kaza, L. C. Yao, P. Bhada-Tata, and F. Van Woerden, *What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050*. World Bank Publications, Aug. 2018.

[8] N. Ferronato, V. Torretta, N. Ferronato, and V. Torretta, "Waste Mismanagement in Developing Countries: A Review of Global Issues," *International Journal of Environmental Research and Public Health*, vol. 16, Mar. 2019.

[9] M. Abdallah, M. Abu Talib, S. Feroz, Q. Nasir, H. Abdalla, and B. Mahfood, "Artificial intelligence applications in solid waste management: A systematic research review," *Waste Management*, vol. 109, pp. 231–246, May 2020.

[10] A. Olugboja and Z. Wang, "Intelligent Waste Classification System Using Deep Learning Convolutional Neural Network," *Procedia Manufacturing*, vol. 35, pp. 607–612, Jan. 2019.

[11] United Nations Department of Economic and Social Affairs, *The Sustainable Development Goals Report 2025*. The Sustainable Development Goals Report, United Nations, July 2025.

[12] Z. Zhang, X. Chen, C. Wang, R. Wang, W. Song, and F. Nie, "Structured multi-view k-means clustering," *Pattern Recognition*, vol. 160, p. 111113, Apr. 2025.

[13] H. N. Vu, M. H. Nguyen, and C. Pham, "Masked face recognition with convolutional neural networks and local binary patterns," *Applied Intelligence*, vol. 52, pp. 5497–5512, Mar. 2022.

[14] H. Hou, S. Ding, C. Gui Cao, X. Xu, L. Guo, and X. Li, "A comprehensive survey of image clustering based on deep learning," *Pattern Recognition*, vol. 172, p. 112590, Apr. 2026.

[15] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, pp. 395–416, Dec. 2007.

[16] M. Soltanolkotabi and E. J. Candés, "A geometric analysis of subspace clustering with outliers," *The Annals of Statistics*, vol. 40, Aug. 2012.

[17] E. Elhamifar and R. Vidal, "Sparse Subspace Clustering: Algorithm, Theory, and Applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 2765–2781, Nov. 2013.

[18] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust Recovery of Subspace Structures by Low-Rank Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 171–184, Jan. 2013.

[19] A. Ng, M. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*, vol. 14, MIT Press, 2001.

[20] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 888–905, Aug. 2000.

[21] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Nov. 2004.

[22] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, June 2005.

[23] R. Arandjelovic and A. Zisserman, "All About VLAD," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1578–1585, June 2013.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015.

[25] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," Apr. 2015.

[26] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," June 2021.

[27] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging Properties in Self-Supervised Vision Transformers," May 2021.

[28] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "DINOv2: Learning Robust Visual Features without Supervision," Feb. 2024.

[29] O. Siméoni, H. V. Vo, M. Seitzer, F. Baldassarre, M. Oquab, C. Jose, V. Khalidov, M. Szafraniec, S. Yi, M. Ramamonjisoa, F. Massa, D. Haziza, L. Wehrstedt, J. Wang, T. Darcet, T. Moutakanni, L. Sentana, C. Roberts, A. Vedaldi, J. Tolan, J. Brandt, C. Couprie, J. Mairal, H. Jégou, P. Labatut, and P. Bojanowski, "DINOv3," Aug. 2025.

[30] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised Deep Embedding for Clustering Analysis," May 2016.

[31] Y. Li, P. Hu, Z. Liu, D. Peng, J. T. Zhou, and X. Peng, "Contrastive Clustering," Sept. 2020.

[32] H. Zhong, J. Wu, C. Chen, J. Huang, M. Deng, L. Nie, Z. Lin, and X.-S. Hua, "Graph Contrastive Clustering," Apr. 2021.

[33] W. V. Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, and L. V. Gool, "SCAN: Learning to Classify Images without Labels," July 2020.

[34] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid, "Deep Subspace Clustering Networks," Sept. 2017.

[35] D. L. Donoho, "High-Dimensional Data Analysis: The Curses and Blessings of Dimensionality," *AMS math challenges lecture*, vol. 1, p. 32, 2000.

[36] "ISO 14001:2015 — Environmental management systems — Requirements with guidance for use," 2015.

[37] "ISO 9001:2015 — Quality management systems — Requirements," 2015.

[38] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, Dec. 2001.

[39] Z. Lin, R. Liu, and Z. Su, "Linearized Alternating Direction Method with Adaptive Penalty for Low-Rank Representation," in *Advances in Neural Information Processing Systems*, vol. 24, Curran Associates, Inc., 2011.

[40] R. Basri and D. Jacobs, "Lambertian reflectance and linear subspaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 218–233, Feb. 2003.

[41] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, Mar. 1982.

[42] J. Delhumeau, P.-H. Gosselin, H. Jégou, and P. Pérez, "Revisiting the VLAD image representation," in *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, (New York, NY, USA), pp. 653–656, Association for Computing Machinery, Oct. 2013.

[43] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image Classification with the Fisher Vector: Theory and Practice," *International Journal of Computer Vision*, vol. 105, pp. 222–245, Dec. 2013.

[44] F. Perronnin, J. Sánchez, and T. Mensink, *Improving the Fisher Kernel for Large-Scale Image Classification*, vol. 6314. Springer, Sept. 2010.

[45] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1615–1630, Oct. 2005.

[46] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, vol. 5.1, pp. 281–298, University of California Press, Jan. 1967.

[47] S. Tripathi, S. K. Singh, and L. H. Kuan, "Bag of Visual Words (BoVW) with Deep Features – Patch Classification Model for Limited Dataset of Breast Tumours," Feb. 2022.

[48] A. Vedaldi and B. Fulkerson, "Vlfeat: An open and portable library of computer vision algorithms," in *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, (New York, NY, USA), pp. 1469–1472, Association for Computing Machinery, Oct. 2010.

[49] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data Via the EM Algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, pp. 1–22, Sept. 1977.

[50] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.