Universidade Estadual de Campinas
Instituto de Computação

Adán Echemendía Montero

A Divide-and-Conquer Clustering Approach based on Optimum-Path Forest

Uma Abordagem de Agrupamento baseada na Técnica de Divisão e Conquista e Floresta de Caminhos Ótimos

CAMPINAS
2018

Adán Echemendía Montero

A Divide-and-Conquer Clustering Approach based on Optimum-Path Forest

Uma Abordagem de Agrupamento baseada na Técnica de Divisão e Conquista e Floresta de Caminhos Ótimos

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. Alexandre Xavier Falcão

Este exemplar corresponde à versão final da Dissertação defendida por Adán Echemendía Montero e orientada pelo Prof. Dr. Alexandre Xavier Falcão.

CAMPINAS
2018

Agência(s) de fomento e nº(s) de processo(s): CAPES

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

Ec43d

Echemendía Montero, Adán, 1988-
A divide-and-conquer clustering approach based on optimum-path forest / Adán Echemendía Montero. – Campinas, SP : [s.n.], 2018.

Orientador: Alexandre Xavier Falcão.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Reconhecimento de padrões. 2. Segmentação de imagens. 3. Floresta de caminhos ótimos. I. Falcão, Alexandre Xavier, 1966-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Uma abordagem de agrupamento baseada na técnica de divisão e conquista e floresta de caminhos ótimos
Palavras-chave em inglês:
Pattern recognition
Image segmentation
Optimum-path forest
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Alexandre Xavier Falcão [Orientador]
João Paulo Papa
Luciano Silva
Data de defesa: 25-05-2018
Programa de Pós-Graduação: Ciência da Computação

Universidade Estadual de Campinas
Instituto de Computação

Adán Echemendía Montero

A Divide-and-Conquer Clustering Approach based on Optimum-Path Forest

Uma Abordagem de Agrupamento baseada na Técnica de Divisão e Conquista e Floresta de Caminhos Ótimos

Banca Examinadora:

- Prof. Dr. Alexandre Xavier Falcão
  UNICAMP
- Prof. Dr. Luciano Silva
  UFPR
- Prof. Dr. Joao Paulo Papa
  UNESP

A ata da defesa com as respectivas assinaturas dos membros da banca encontra-se no processo de vida acadêmica do aluno.

Campinas, 25 de maio de 2018

Acknowledgements

I would like first to thank my wife, my parents, my grandmother, and my parents-in-law for providing me with unfailing support and continuous encouragement during this period of my life. This accomplishment would not have been possible without them.

I would like to thank my thesis advisor Prof. Alexandre Xavier Falcão for his attention, patience, guidance, and support whenever I ran into troubles or had questions about my research or writing.

I would like to acknowledge the laboratory fellows of both the LIV and the LIDS for the assistance provided in these years, with special thanks to Alan Peixinho, Samuel Martins, and Anderson Carlos.

I would also like to thank the rest of my friends, who in some way contributed to the person I am today. I would like to acknowledge Jose Carlos Urra and Camila Gonzalez, who helped me with the review of this text.

Finally, I must express my very profound gratitude to Brazil, Unicamp, and CAPES/CNPq agencies which gave me this opportunity and made this work possible.

Resumo

O agrupamento de dados é um dos principais desafios em problemas de Ciência de Dados. Apesar do seu progresso científico em quase um século de existência, algoritmos de agrupamento ainda falham na identificação de grupos (clusters) naturalmente relacionados com a semântica do problema. Ademais, os avanços das tecnologias de aquisição, comunicação, e armazenamento de dados acrescentam desafios cruciais com o aumento considerável de dados, os quais não são tratados pela maioria das técnicas. Essas questões são endereçadas neste trabalho através da proposta de uma abordagem de divisão e conquista para uma técnica de agrupamento única em encontrar um grupo por domo da função de densidade de probabilidade dos dados — o algoritmo de agrupamento por floresta de caminhos ótimos (OPF - Optimum-Path Forest). Nesta técnica, amostras são interpretadas como nós de um grafo cujos arcos conectam os k-vizinhos mais próximos no espaço de características. Os nós são ponderados pela sua densidade de probabilidade e um mapa de conexidade é maximizado de modo que cada máximo da função densidade de probabilidade se torna a raiz de uma árvore de caminhos ótimos (grupo). O melhor valor de k é estimado por otimização em um intervalo de valores dependente da aplicação. O problema com este método é que um número alto de amostras torna o algoritmo inviável, devido ao espaço de memória necessário para armazenar o grafo e o tempo computacional para encontrar o melhor valor de k. Visto que as soluções existentes levam a resultados ineficazes, este trabalho revisita o problema através da proposta de uma abordagem de divisão e conquista com dois níveis. No primeiro nível, o conjunto de dados é dividido em subconjuntos (blocos) menores e as amostras pertencentes a cada bloco são agrupadas pelo algoritmo OPF. Em seguida, as amostras representativas de cada grupo (mais especificamente as raízes da floresta de caminhos ótimos) são levadas ao segundo nível, onde elas são agrupadas novamente. Finalmente, os rótulos de grupo obtidos no segundo nível são transferidos para todas as amostras do conjunto de dados através de seus representantes do primeiro nível. Nesta abordagem, todas as amostras, ou pelo menos muitas delas, podem ser usadas no processo de aprendizado não supervisionado, sem afetar a eficácia do agrupamento e, portanto, o procedimento é menos susceptível a perda de informação relevante ao agrupamento. Os resultados mostram agrupamentos satisfatórios em dois cenários, segmentação de imagem e agrupamento de dados arbitrários, tendo como base a comparação com abordagens populares. No primeiro cenário, a abordagem proposta atinge os melhores resultados em todas as bases de imagem testadas. No segundo cenário, os resultados são similares aos obtidos por uma versão otimizada do método original de agrupamento por floresta de caminhos ótimos.

Palavras-chave: agrupamento, floresta de caminhos ótimos, segmentação de imagem, transformada imagem-floresta, paradigma de divisão e conquista, aprendizado de máquina.

Abstract

Data clustering is one of the main challenges when solving Data Science problems. Despite its progress over almost one century of research, clustering algorithms still fail in identifying groups naturally related to the semantics of the problem. Moreover, the advances in data acquisition, communication, and storage technologies add crucial challenges with a considerable data increase, which are not handled by most techniques. We address these issues by proposing a divide-and-conquer approach to a clustering technique, which is unique in finding one group per dome of the probability density function of the data — the Optimum-Path Forest (OPF) clustering algorithm. In the OPF-clustering technique, samples are taken as nodes of a graph whose arcs connect the k-nearest neighbors in the feature space. The nodes are weighted by their probability density values and a connectivity map is maximized such that each maximum of the probability density function becomes the root of an optimum-path tree (cluster). The best value of k is estimated by optimization within an application-specific interval of values. The problem with this method is that a high number of samples makes the algorithm prohibitive, due to the required memory space to store the graph and the computational time to obtain the clusters for the best value of k. Since the existing solutions lead to ineffective results, we decided to revisit the problem by proposing a two-level divide-and-conquer approach. At the first level, the dataset is divided into smaller subsets (blocks) and the samples belonging to each block are grouped by the OPF algorithm. Then, the representative samples (more specifically the roots of the optimum-path forest) are taken to a second level where they are clustered again. Finally, the group labels obtained in the second level are transferred to all samples of the dataset through their representatives of the first level. With this approach, we can use all samples, or at least many samples, in the unsupervised learning process without affecting the grouping performance and, therefore, the procedure is less likely to lose relevant grouping information. We show that our proposal can obtain satisfactory results in two scenarios, image segmentation and the general data clustering problem, in comparison with some popular baselines. In the first scenario, our technique achieves better results than the others in all tested image databases. In the second scenario, it obtains outcomes similar to an optimized version of the traditional OPF-clustering algorithm.

Keywords: clustering, optimum-path forest, image segmentation, image foresting transform, divide-and-conquer paradigm, machine learning.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

A dataset may consist of samples from a given problem mathematically represented by a set of measures per sample, usually referred to as feature vector or point in some n-dimensional feature space. For instance, samples may be images or image elements, such as pixels, regions, and objects, whose representation can be based on color, texture, and shape measures depending on the case. The dissimilarity between samples can be measured by a distance function between their feature vectors.

Data Clustering is the problem of finding meaningful groups of similar samples according to the distance function and mathematical representation. It is a well-known problem with large numbers of contributions and applications [59, 4]. According to [4], the clustering problem can be thought of as either an exploratory (descriptive) task or a pre-processing step. In the former, the objectives are discovery and exploitation of hidden patterns in the data. In the latter, it aims to facilitate another data mining or machine learning task.

The samples are very often drawn from a set of possible categories (classes) related to the problem. In this case, the clustering algorithm should be able to learn a grouping model that can minimize the number of samples from distinct classes in the same cluster, while keeping the number of groups as small as possible (since the trivial solution is to consider each sample a distinct group). Given that the design of such grouping model must be done with no category information about the samples, the problem is referred to as *unsupervised learning*. The model should also be able to propagate group labels to new samples with a minimum mixture of classes per group. Note, however, this is an ill-posed problem since there is no guarantee that the labeling obtained by a clustering algorithm is the same as the true-class labeling — i.e., samples from multiple classes may fall in the same group, or many groups can be associated with the same class. Additional class information is then required, at least the classes of the representative samples, such that the remaining samples from the same group can be classified in the same class.

This MSc thesis focus on clustering algorithms based on the *Optimum-Path Forest* (OPF) framework [94, 102]. This framework interprets the training set of a dataset as a graph, whose nodes are the samples and arcs are defined by an *adjacency relation* between samples. The intention is to explore the "strength of connectedness" between samples in the feature space, as defined by a *connectivity function* (path-value function), for data clustering and pattern classification. This idea stems from a previous framework, named

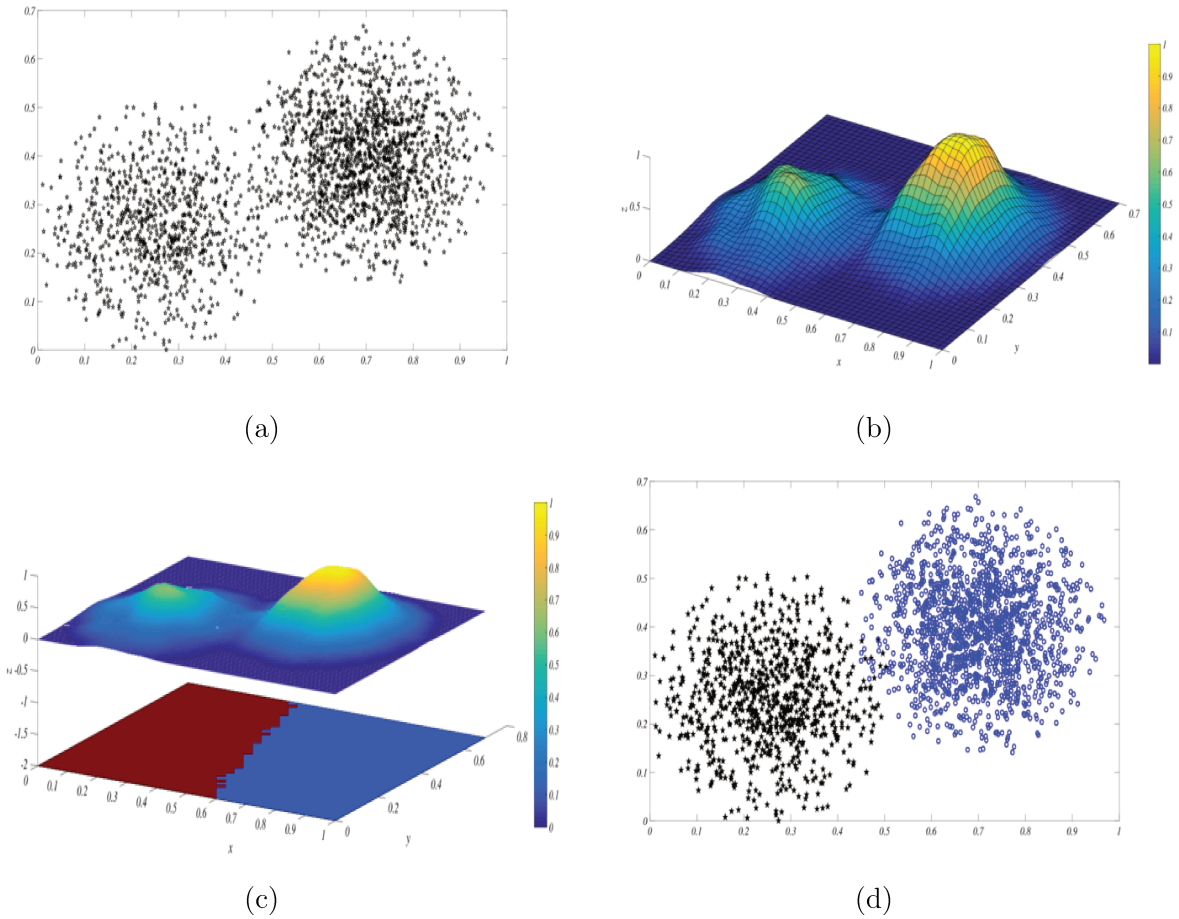
Image Foresting Transform (IFT) [44], for the design of image processing and analysis operators based on connectivity between image elements, such as pixels, regions, and pixel vertices. OPF essentially uses the same algorithm to extend the IFT operators for clustering and classification. In OPF, a grouping model or a pattern classifier is an optimum-path forest in the input graph. The group/class label assignment to new samples is performed by finding the root of the forest which would offer an optimum path to the new sample, as though that sample were part of the training set, and assigning the label of that root.

The OPF-clustering method [102] can identify natural groups as domes of a *probability density function* (PDF) estimated from the training samples (see Figure 1.1). The graph nodes are these samples and their k-nearest neighbors in the feature space form the arcs. Such domes of the PDF represent high concentrations of samples at some scale of the problem. For instance, by assuming an observer at a very far distance, any training set becomes a single group of points in the feature space. A distinction among groups of samples appears as the observer gets closer to the training set. The choice of such scale is obtained by optimization within a finite search space as defined by the degree of the nodes in the graph (i.e., the value of k). The OPF algorithm finds one representative sample per maximum of the PDF and outputs an optimum-path forest rooted at those representatives, such that each cluster is an optimum-path tree. That is, each root defines a cluster by conquering the "most strongly connected" samples according to the given path-value function. In this way, the training forest becomes a classifier that can assign to any new sample the label of its most strongly connected root. This method can handle plateaus of maximum (by electing a single root per maximum), some overlapping among clusters, and groups with arbitrary shapes.

1.1 Motivation

The OPF-clustering method can be very effective when using all data to construct the forest, but it becomes prohibitive (in required memory space and processing time) as samples and features per sample increase in number. Given that the sizes of the datasets have grown large very rapidly, due to new technologies, it is crucial to maintain the OPF-clustering method viable with no loss in effectiveness. For example, imaging devices, such as digital cameras and tomographic scanners, can acquire images with millions of pixels for classification (segmentation).

For large datasets, the authors in [102] suggest the use of a small training set composed of randomly selected samples. Such an unsupervised training set allows a fast construction of the weighted k-nearest neighbor graph followed by an efficient procedure of group label propagation to the remaining samples in the dataset. This technique has already succeeded with training sets of about 400 voxels when classifying gray matter, white matter, and cerebral spinal fluid in magnetic resonance images of the brain (images with about 1.5 million voxels) [23]. However, the result of the grouping may be compromised in some cases where relevant information is lost in the sampling process (when the training set is formed). In this thesis, we name *OPF-Large-Data* this variant of the OPF-clustering



(a) (b)

(c) (d)

Figure 1.1: (a) A two-dimensional space where the samples are represented by points. (b) The density value of each sample is displayed as a 3D surface. (c) The groups are defined by the maxima of the PDF, whose influence areas are illustrated on the 2D projection by regions of different colors. (d) Samples receive the label of their most strongly connected maximum.

method to facilitate its reference in the text. The authors in [102] have also proposed to constrain the k-nearest neighbors within a spatial adjacency of each pixel for the purpose of image segmentation. However, this approach over-segments the image making difficult to control the number of clusters.

In view of that, we address the limitations of the OPF-clustering method and its variants for large datasets.

1.2 Objectives

There are two important questions to be answered in this MSc thesis:

1. Can the divide-and-conquer paradigm be used to maintain the high effectiveness of the OPF-clustering method, making it viable for large datasets?

2. How can we exploit the divide-and-conquer paradigm to find natural groups by OPF clustering in large, structured (such as images) and unstructured, datasets?

The challenge in the first question is to maintain effectiveness when dividing the dataset into parts, applying OPF clustering in each part, and combining the groups from each part into a final clustering result. For the second question, structured data such as images allow exploiting spatial pixel information, which may lead to different solutions for image segmentation. We wish to use spatial information without over-segmenting the image. We are also interested in further improving and exploiting the OPF-Large-Data approach.

Image segmentation is one of the most fundamental and challenging problems in Image Processing and Computer Vision. Clustering pixels into relevant objects and background is extremely hard without the user input (interactive segmentation) or a well-controlled situation with specific information about the application (automatic segmentation). We intend to evaluate the proposed approach in this context, separating a given object from the background in images of different application domains (natural, biological, and medical). In our case, we count with the ground-truth segmentation of those images, so we can easily validate the results obtained by our technique when comparing them with the desired outputs. A strategy is to measure the ability of the method to represent the object as the union of its internal clusters when trying to reduce as much as possible the number of clusters. We also assess the methods for arbitrary data clustering from other applications.

1.3 Main contributions

This work proposes a two-level divide-and-conquer OPF-clustering approach suitable for large datasets. The pipeline of the technique is explained in Figure 1.2. At the first level, the data (Figure 1.2a) is divided into blocks and the samples of each block are clustered separately using the OPF algorithm (Figure 1.2b). Then, the representative samples (one per cluster and, more specifically, the root of each optimum-path tree) are taken to the second level (Figure 1.2c) to be clustered again using the OPF algorithm (Figure 1.2d). Finally, the group labels obtained in the second level are transferred to all samples (the samples that form the first level) through their representatives in the first level (Figure 1.2e) resulting in the final partition (Figure 1.2f). We name this technique *OPF-Blocks-2* because of its two levels.

The size of the blocks in the first level and the number of samples in the second level must be neither too large nor too small. As long as each block has a sufficient number of samples to represent the problem, this should not compromise the performance of the method. However, we noticed that in the case of 2D and 3D images, or other very large datasets, a small number of bigger blocks is preferred as long as a suitable sampling strategy is applied to select one training set per block for clustering followed by group label propagation — i.e., the optimum-path forest of each training set is extended to the remaining samples of the block. At this point, our solution for each block is similar to the OPF-Large-Data approach. The difference is that we use a more effective grid sampling technique, in the case of image segmentation, to compose the training sets in each block of the image. We have also improved efficiency in the estimation of the parameter k

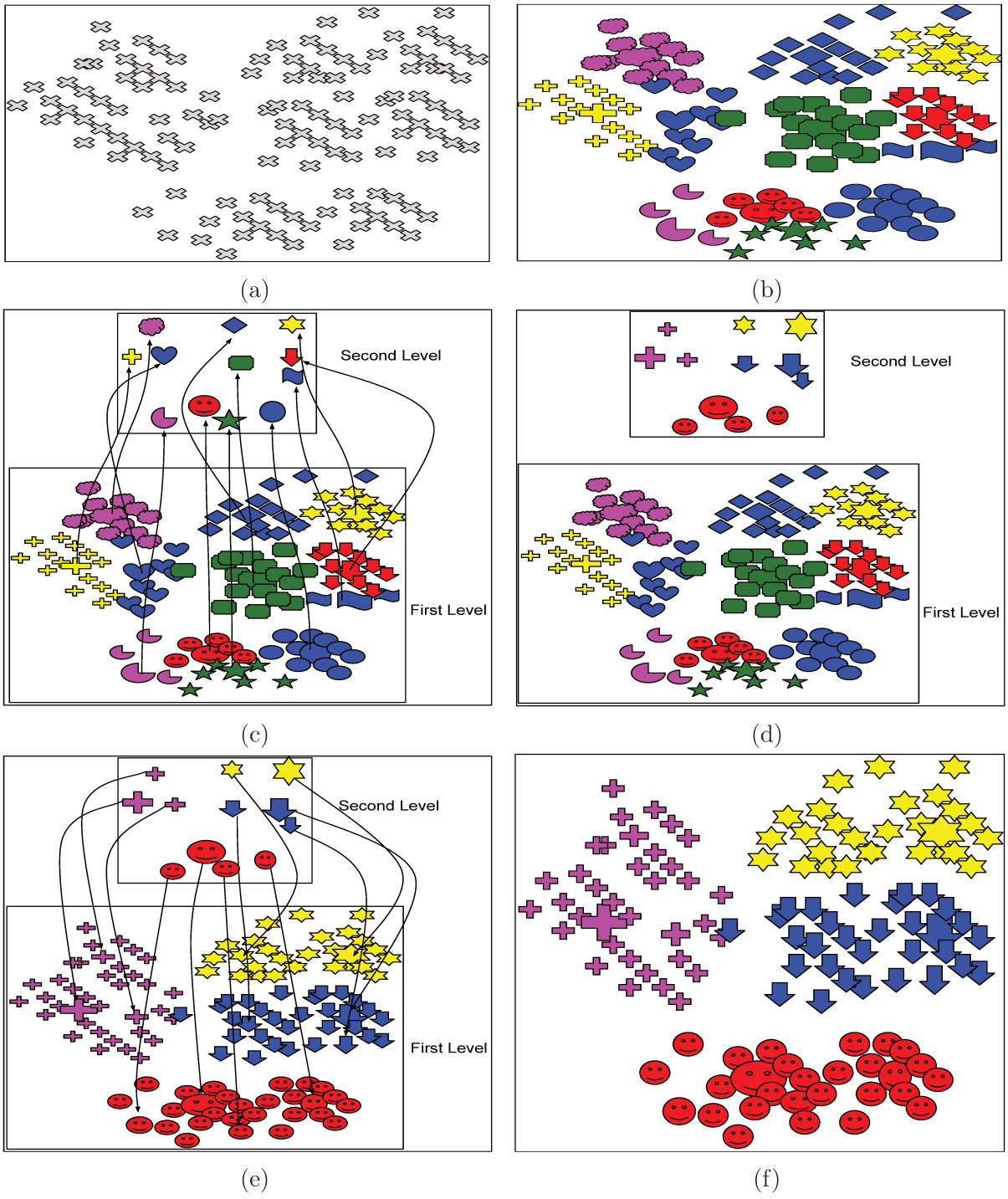


Figure 1.2: Clustering a toy dataset with the proposed divide-and-conquer algorithm. (a) Some unlabeled data. (b) The data is divided into blocks with random sampling (each color indicates a different block) and each block is clustered with the OPF algorithm (each shape of the same color indicates a different cluster in the same block, e.g., the block marked with the blue color is partitioned into four groups: heart, oval, diamond, and wave). The prototypes of the groups are indicated with larger shapes. (c) The prototypes of the clusters in the first level are promoted to the second level (summarized dataset). (d) The samples in the second level are also clustered with the OPF algorithm (each group is distinguished by a different color and shape). (e) The group labels obtained in the second level are propagated to all data (first level) by means of their prototypes. (f) The dataset is partitioned into four groups.

without losing effectiveness. By dividing the image into blocks, we are actually using a considerably higher number of training samples (the union of the training sets of each block) than OPF-Large-Data, which avoids loss of relevant data information for clustering.

1.4 Organization of the text

This MSc thesis is organized as follows. Chapter 2 summarizes the main concepts and techniques related to the clustering task. Chapter 3 presents essential information to understand the methods in this work, including the image segmentation process and the OPF-clustering technique. Chapter 4 details the proposed approach, highlighting how it works in the two tested scenarios, image segmentation and arbitrary data clustering. Chapter 5 describes the experiments and discusses the obtained results. Finally, Chapter 6 presents the conclusions and provides directions for future work.

Chapter 2

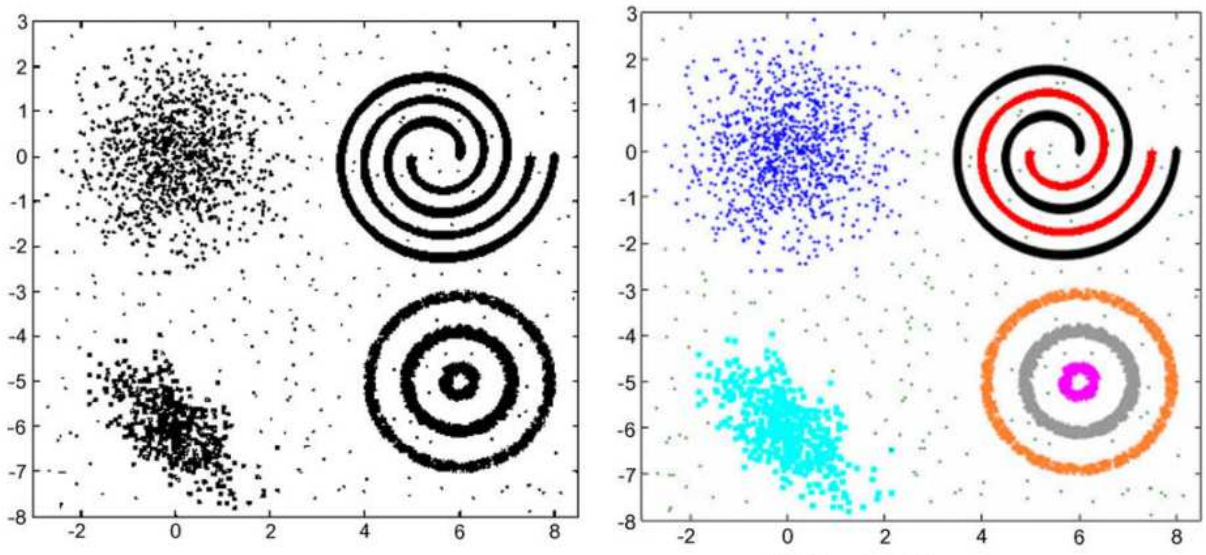
Clustering Overview

Recall that the main objectives in this master thesis are to extend the OPF-based clustering technique [102] through a divide-and-conquer strategy and to evaluate this new approach in different scenarios. In this chapter, we introduce the basic notions related to the clustering task, some common applications of the grouping methods, and a taxonomy of the clustering algorithms referring to some classical and cutting-edge approaches.

2.1 Cluster analysis

Cluster analysis is the organization of a collection of patterns (samples), usually represented as a vector of measurements or a point in a d-dimensional space $\Re^d$, into clusters based on similarity. Intuitively, samples within a valid cluster are more similar to each other than to those that belong to a different cluster. Figure 2.1 shows an example of the variety of possible groups in the same dataset. The goal is to develop an automatic algorithm to discover the natural grouping (Figure 2.1b) in the unlabeled data (Figure 2.1a). Figure 2.1 conveys the idea that clusters can differ in terms of their shape, size, and density. Furthermore, the detection of natural groups can be even more difficult if the data contains noise. An ideal cluster can be defined as a set of samples that is compact and isolated. Actually, a cluster is a subjective entity whose significance and interpretation requires domain knowledge.

It is important to understand the differences between *clustering* (*unsupervised learning*) and *classification* (*supervised learning*) in the context of Pattern Recognition. In supervised classification, the data is a collection of *labeled samples* and the problem is to label a newly encountered, yet unlabeled, sample. With these techniques, the labeled samples (*training samples*) can be used to learn a representation and a decision model for the existing *classes*, which in turn are used to label a new sample. In the case of clustering, there is no such pre-annotated data and the problem can be to find the representation and group a given collection of unlabeled samples into *meaningful clusters*. In this thesis, we assume the sample representation is chosen *a priori* and focus on the grouping problem only. The clusters are subjective entities that make sense to a specific application. In some way, the *group labels* are associated with classes also, but the group labels are data-driven — i.e., they are obtained solely from the data. Therefore, one class



(a) Input data (b) Desired clustering

Figure 2.1: A representation of the variety of possible groups in the same dataset. (a) Some unlabeled data. (b) Seven different groups (marked by different colors) differ in density, shape, and size. It is very likely that almost none or none of the available clustering algorithms can detect all these clusters in the same input data. However, these patterns can be easily discovered by a human without much effort. This figure was obtained from [59].

may be represented by one or multiple groups.

Many fields of the Sciences and Engineering employ clustering techniques. Image segmentation, an important problem in Image Processing and Computer Vision, can be formulated as a clustering problem [111]. Clustering of documents is a common practice to generate hierarchies of topics for effective information access [104] and retrieval [19]. These unsupervised techniques are also useful to divide customers into different categories for efficient marketing [11], to study genome data [14], to address the problems of face recognition and identification [137, 55], to summarize video [88], among other applications. Summing up, the techniques of data clustering have been used for the following three main purposes [59].

- **Understanding the underlying data structure:** to get information about the data, generate hypotheses, detect anomalies, and identify key features.
- **Natural classification:** to identify the degree of similarity among different kinds of documents, objects, and patterns in general. Examples are the determination of similar snippets of music and similar photographs.
- **Compression**: to organize the data and summarize it through cluster prototypes. Data summarization can be helpful in creating compact data representations, which are easier to process and interpret in a wide variety of applications.

Traditional clustering activity involves the following steps [60]: (1) feature engineering, (2) definition of a similarity measure between samples, (3) grouping, (4) data abstraction (if needed), and (5) evaluation of the output (if needed). Figure 2.2 displays the sequence

of the first three steps, considering a feedback path where the clustering outcome can act on the feature descriptor and the similarity calculations.

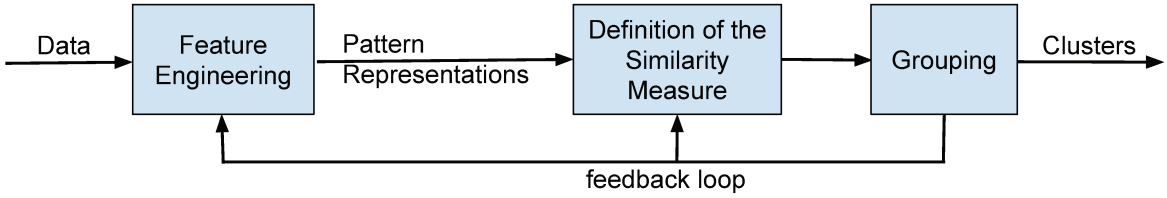


Figure 2.2: Some common stages in clustering.

Feature engineering is the process of determining the most effective *descriptor* (representation) for the data according to the problem to be solved. Although this process usually makes use of domain knowledge incorporated by specialists, it is often helped by visualization methods, feature selection procedures, and dimensionality reduction techniques as well. *Dimensionality reduction methods* are used to remove irrelevant and redundant attributes by projecting the original features on to a new space with lower dimensionality. Some examples of these techniques include Principle Component Analysis (PCA) and Singular Value Decomposition (SVD). t-Distributed Stochastic Neighbor Embedding (t-SNE) [75] is a non-linear dimensionality reduction algorithm that is well-suited for visualizing high dimensional data into a two or three-dimensional space. In this case, the projection coordinates present no mathematical relation to the original feature space. This technique tries to keep both, the original local and global structure of the samples, in the lower space, which makes it very useful when evaluating sample descriptors for a learning problem. *Feature selection approaches* target to find a subset of attributes that minimize redundancy and maximize relevance to the learning objective. Information Gain, Lasso, and Relief are some popular feature selection techniques. Recently, there is a tendency to use *automatic feature engineering techniques* (e.g., deep learning techniques) which have been shown to be more effective than knowledge-based approaches in many problems [81].

Pattern proximity is a fundamental concept to define groups in the data. Since it may exist a variety of feature types and scales in the representations of the samples, this measure should be chosen carefully. A common way to calculate the dissimilarity between two samples is using a *distance metric* on the *feature space*. The most popular metric for continuous features is the *Euclidean distance*, which is a special case ($p = 2$) of the Minkowski metric

$$d_p(x_i, x_j) = (\sum_{k=1}^{d} |x_{i,k} - x_{j,k}|^p)^{1/p} = ||x_i - x_j||_p \tag{2.1}$$

where $(x_{i,1}, ..., x_{i,d})$ is the descriptor (or *feature vector*) of the sample x_i. Some other known metrics are the cosine similarity, the Mahalanobis distance, and the Gaussian distance.

The operation of extracting a simple and compact representation from a dataset is

known as *data abstraction* or *data summarization*. In many cases, this summarized representation is very helpful for subsequent automatic analysis by a machine, or an interpretation by a data analyst. In the clustering context, data abstraction refers to derive a summarized description of each cluster, in terms of *cluster prototypes* or *representatives* — e.g., the centroids of the clusters.

There are many ways in which a clustering result can be evaluated. The analysis can be done through external knowledge-based supervision, interactive visualization by a specialist, comparison and combination of multiple solutions to evaluate different possibilities, among other options. *Cluster validation* embraces three main tasks [134]: clustering effectiveness (it assesses the goodness of the clustering according to the needs of the application), clustering stability (it assesses the sensitivity of the clustering results to parameter variations, e.g., the number of clusters), and clustering tendency (it assesses the convenience of applying a clustering solution to a problem, e.g., if the data has any inherent grouping structure). Most existing metrics to address these tasks can be classified into:

External: External validation metrics use some expert-specified knowledge about the clusters that are not inherent to the data — e.g., the real partition of the data. Such external information is not available in many real-world applications; however, these metrics allow to validate grouping methods in some synthetic and classification datasets.

Internal: Internal validation metrics use criteria derived from the data itself. They utilize notions of intra-cluster similarity (compactness) contrasted with notions of inter-cluster separation, that usually results in a trade-off between maximizing these two goals.

Relative: Relative validation metrics are used to compare different outcomes of the same clustering algorithm by different parameter settings — e.g., varying the number of desired clusters k.

Different approaches to data grouping can be described with the help of the hierarchy displayed in Figure 2.3, although other taxonomic representations are also possible. This representation is mainly based on the book [134]. Some clustering techniques can be associated with more than one category as we shall see. We add a special category called "Large Datasets" in reference to the methods designed with the main objective of clustering large datasets. At the top level of the taxonomy, there is a distinction among the clustering approaches with respect to their strategy (based on cluster representatives, hierarchy, density, graph, and for large datasets). The next sections explain some relevant concepts about each category alluding to some widely used algorithms.

2.2 Representative-based clustering

Given a dataset $N = \{x_1, ..., x_n\}$ with n samples in a d-dimensional space, the goal of any clustering technique is to discover the natural grouping $C = \{C_1, C_2, ..., C_k\}$ such that $x \in C_i$, $\forall x \in N$ and $i \in [1, k]$, $\cap_{i=1}^{k} C_i = \emptyset$, and $\cup_{i=1}^{k} C_i = N$. In *representative-based clustering methods*, it is necessary to have a representative (prototype) sample for each cluster C_i that summarizes the group. A common choice for this prototype is the mean

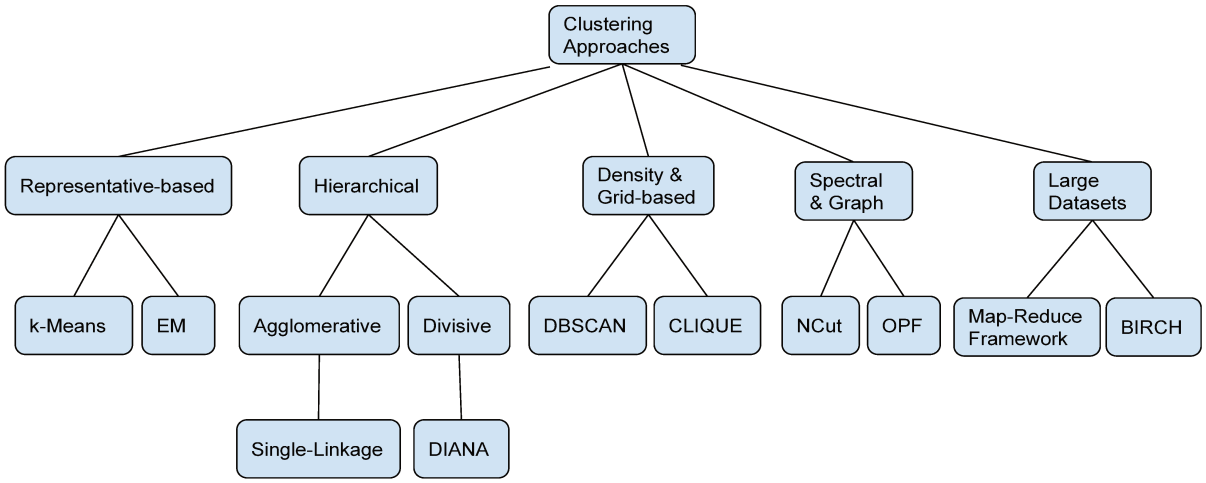


Figure 2.3: A taxonomy of the clustering techniques.

(also called the centroid) of all samples in the cluster

$$\mu_i = \frac{1}{n_i} \sum_{x \in C_i} x \tag{2.2}$$

where $n_i = |C_i|$ is the number of samples in cluster C_i.

k-Means is the best-known representative-based clustering algorithm. It starts by choosing k random samples as the initial centroids. Then, each sample is associated with the closest centroid based on a particular proximity measure. Once the groups are formed, the centroids of each cluster are updated according to Equation 2.2. The algorithm iteratively repeats these two steps until the centroids do not move significantly or any other relaxed convergence criterion is met. k-Means is essentially an optimization problem, with the goal of minimizing the Sum of Squared Errors (SSE) objective function, that converges to a local minimum [4]. The SSE is given by

$$SEE(C) = \sum_{i=1}^{k} \sum_{x \in C_i} \| x - \mu_i \|^2 \tag{2.3}$$

where μ_i is the centroid of the cluster C_i (see Equation 2.2).

A wide range of similarity measures can be used with k-Means, however, the Euclidean distance is the most popular choice. The parameter k specifies the desired number of groups to be returned by the technique. A problem is that in many cases this value is unknown.

Many optimizations have been proposed to address the two major factors that impact the performance of k-Means: choosing the initial centroids and estimating the number of clusters k. To try to solve the second issue, [15] suggests an extension of k-Means, called ISODATA, that merges and divides groups dynamically after an initial clustering. In ISODATA, groups are merged if their distance is less than a threshold or if they have fewer than a certain number of samples. In the same way, a cluster is split if its standard deviation surpasses an user-defined threshold.

The k-Means procedure is an example of a *hard assignment* clustering technique, where each sample can belong only to one group. There are other approaches, such as Fuzzy C-Means [18], that extend this idea to consider a *soft assignment* result — i.e., each sample has a degree of membership with each discovered cluster.

Mixture Models [78] have been addressed in many ways to support clustering problems. The underlying assumption is that the samples are drawn from one of several components and the problem is how to estimate the parameters of each component that best fit the data. Recognizing which component generates each sample produces a clustering of the data. Gaussian Mixture Model (GMM) is the most well-known mixture model, where each component is a Gaussian distribution. This model has been widely used for clustering in many applications, such as speaker identification and verification [99, 100], image segmentation [97], and object tracking [114]. One efficient way of discovering the number of components in a GMM is using the Bayesian Information Criterion (BIC) [108]. In theory, assuming that data are abundant and actually generated from a mixture of Gaussian distributions, this criterion recovers the true number of components. The biggest problem in learning Gaussian Mixture Models from unlabeled data is finding which samples come from which component. Expectation-Maximization (EM) [32] is a well-founded statistical algorithm that gets around this difficulty by an iterative process. First, it initializes the model with random components (or with another heuristic initialization) and computes the probability that each sample is generated by each of these components. Then, it tweaks the parameters of the model to maximize the likelihood of the data given those assignments. Repeating this process, EM always converges to a local optimum.

Representative-based clustering algorithms usually try to discover the natural grouping in the data by optimizing a specific objective function and gradually improving the result in an iterative process [4]. They are effective to detect compact spherical-shaped clusters but can fail with other structures. These methods generally require some user information as the number of desired groups. A problem is that this information is mostly unknown and must be estimated.

2.3 Hierarchical clustering

Hierarchical clustering algorithms overcome some of the disadvantages associated with the representative-based methods. These techniques create a hierarchy of clusters (cluster dendrogram) organizing the data into different levels of granularity. With them, a specialist can tune up the clustering result by splitting up the dendrogram into different levels without re-running the algorithm. Figure 2.4 shows an example of a hierarchical clustering of five labeled points.

Hierarchical clustering can be done in either bottom-up (agglomerative) way or top-down (divisive) way.

Agglomerative: These clustering techniques start by taking singleton clusters (clusters containing only one sample) at the bottom level and continue merging "the two more similar clusters" at a time to build a bottom-up hierarchy. A variety of choices are possible in terms of how to measure this similarity. Some options are single-linkage (nearest neigh-

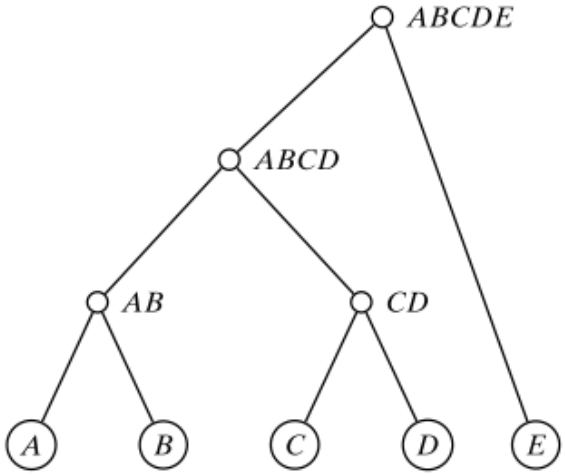


Figure 2.4: Dendrogram as a result of a hierarchical clustering of five labeled points: A, B, C, D, and E.

bor), complete-linkage (diameter), all-pairs linkage (average linkage), centroid-linkage, and Ward's criterion (minimum variance). In single-linkage [79], the two clusters with the shortest distance between any pair of corresponding samples are combined. Because of its local behavior, single-linkage is capable of effectively perceive groups of samples with non-elliptical and elongated forms. However, it is sensitive to noise and outliers in the data. For the complete-linkage [67], the distance to consider between two clusters is the maximum of all pairwise distances between their samples (the distance between the most dissimilar samples). As this method takes the cluster structure into consideration, it generally recovers compact shaped clusters. However, this technique is also sensitive to outliers as single-linkage. In all-pairs linkage, the dissimilarity between two groups is the average distance over all pairs of the corresponding samples, whereas, in centroid-linkage, it is the distance between the two centroids. Ward's criterion [127] uses the SSE (see Equation 2.3) to determine the distance between groups, and its objective is to merge the two clusters that minimize the total within-cluster variance — i.e., the two clusters that together have the smallest value of SSE.

Divisive: These grouping methods, on the other hand, start from all the samples in a huge macro-cluster and continuously split a cluster into two, generating a top-down hierarchy of groups. They can be considered global approaches since they have the complete information before splitting the data. These techniques have the advantage of being more efficient as compared to the agglomerative ones since there is no need to generate the entire hierarchy. Divisive partitioning allows greater flexibility in terms of both the hierarchical structure and balancing level of the different clusters. For example, if the objective is to maintain a balanced tree at each level of the clustering, then the largest cluster can be chosen preferably for the division. METIS [64] uses this last idea in order to create well-balanced clusters in large social networks, where the problem of cluster imbalance is particularly severe. DIANA [66] is a divisive clustering algorithm, where the group with the largest diameter is divided at each step. To split the selected cluster, the algorithm looks first for the most dissimilar sample — i.e., the sample with the greatest

average dissimilarity with the other samples of the same group —, then creates a new group with this sample, and finally reassigns the samples that are closer to the new group than to the old group.

The hierarchical clustering methods are very effective in capturing convexly shaped groups. The main advantage of having a cluster dendrogram is the possibility of cutting the hierarchical tree at any level to obtain the corresponding clusters. This peculiarity allows to execute these techniques without knowing the true number of groups in the data and even try many solutions easily. Also, this kind of clustering can help to visualize and summarize the data. Despite their benefits, these methods are not recommended for large datasets because of their quadratic complexity (k-Means is better in this case), and in general, they are not flexible in the sense that they cannot undo the mergers or divisions once they are made. Agglomerative methods, especially single-linkage, tend to suffer from the chaining problem[1] and are ineffective at capturing arbitrarily shaped clusters. To address the last drawback, techniques such as CURE [53] and CHAMELEON [63] have been proposed in the literature.

2.4 Density-based and grid-based clustering

The representative-based and hierarchical clustering methods are suitable for finding ellipsoid-shaped and other convexly shaped clusters. However, these methods have difficulties to discover non-convex clusters, such as those shown in Figure 2.1, because two samples from different clusters may be closer than two samples in the same cluster. The *density-based methods*, on the other hand, are capable of extracting such clusters and are good at eliminating noise and detecting outliers. They can be considered non-parametric techniques because they do not make any assumptions about the number of clusters or their distribution. Density-based groups are dense areas in the data space divided from each other by sparser areas. Due to their local nature, dense areas in the data can have an arbitrary shape.

DBSCAN [40] is one of the most popular density-based clustering methods. It estimates the density by counting the number of samples within a fixed-radius ϵ-neighborhood and considers two samples as connected if they lie within each other's neighborhood. The central idea of the method is the classification of the samples as core samples, density-reachable samples, and outliers. A sample is called a core sample if its neighborhood of radius ϵ contains at least a minimum number of samples min_samp — i.e., a core sample is in an area of high density. A sample q is directly density-reachable from a core sample p if q is within the ϵ-neighborhood of p. Density-reachability is given by the transitive closure of direct density-reachability. Two samples p and q are called density-connected if there is a third sample o from which both p and q are density-reachable. A cluster is a set of nearby core samples and a set of non-core samples that are density-reachable from a core sample. Any sample that is not a core sample nor a density-reachable sample from a core sample is considered an outlier by the algorithm. Higher values of min_samp

[1] The chaining effect is caused by a small number of noisy data joining sets of samples that should form separate groups.

or lower values of ϵ indicate a higher density needed to form a cluster. A cluster can be detected by recursively taking a core sample, finding all of its neighbors, and so on. The worst-case complexity of DBSCAN is $O(|N|^2)$ when the dimensionality of the data is high. Figure 2.5 displays the clustering result of DBSCAN in a toy dataset highlighting the different types of samples (core samples, density-reachable samples, and outliers) in the data.

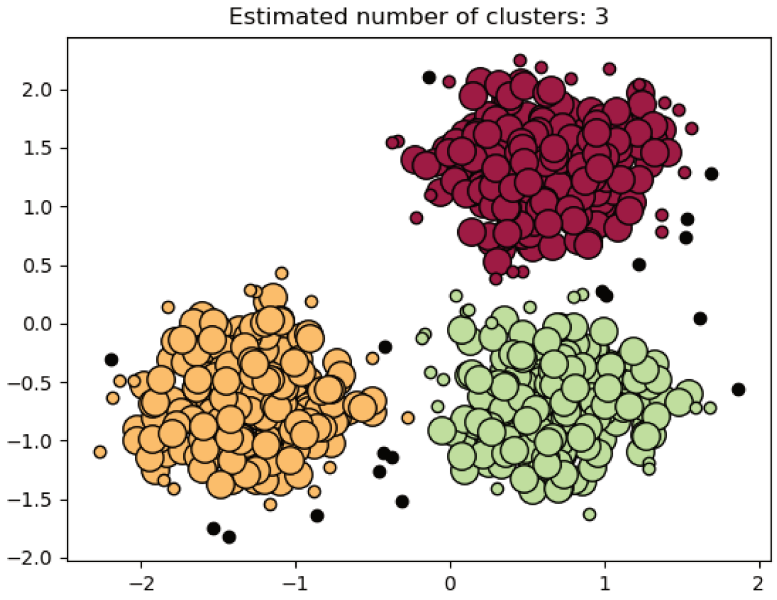


Figure 2.5: Clustering result of DBSCAN in a toy dataset. Larger circles indicate core samples and smaller ones indicate density-reachable samples. The outliers are indicated by black points. This figure was taken from `http://scikit-learn.org/stable/modules/clustering.html`.

DENCLUE [56] generalizes the notion of groups based on density in another way. It is based on the concept of influence functions that mathematically model the effect of a sample in its neighborhood — i.e., the density of each sample is estimated by the sum of the influences of all data samples —, and the clusters are determined by identifying density-attractors (local maxima of the estimated density function). Typical examples of influence functions are square wave functions and Gaussian functions. The OPTICS algorithm [10], unlike DBSCAN, can discover groups of different densities in a dataset, but it only visualizes the cluster structure without actually determining the groups. In [39], the authors propose a procedure based on shared nearest neighbors (SNN) to detect groups of different densities.

Mean-Shift [25] is a popular non-parametric clustering technique which has been used in many areas of Pattern Recognition and Computer Vision. The primary goal of this algorithm is to determine the local maxima (modes) present in the data distribution. Inspired by the kernel density estimation via the Parzen-window, each sample performs a gradient ascent procedure until convergence. As the mean-shift vector always points toward the direction of maximum increase in the density, it can define a path ending in a stationary point (local maxima) of the estimated density. In this way, each maximum

(mode) defines an influence zone (cluster) formed by the samples that reach it. Mean-Shift does not assume any shape on the data clusters and automatically finds the number of groups. The algorithm only depends on a single parameter called *bandwidth* (or *window size*) which dictates the size of the search region to compute the density. However, the selection of the *window size* is not a trivial operation and inappropriate values can produce unexpected results. Another problem with Mean-Shift is that it can fragment a cluster if a maximum is represented by some neighboring points with the same density value. And the reason is that the method does not force a single representative per maximum — i.e., two samples, that should be grouped together, can reach two different points in the same maximum.

Grid-based methods are a specific class of density-based methods designed for mining large multidimensional datasets. These techniques divide the data space into a finite number of cells (creating a grid structure) and then try to discover clusters from dense regions in the cells. Grids were initially proposed in [128], but they only gained popularity after the methods STING and CLIQUE were introduced. The main advantage of grid-based clustering is a significant reduction in execution time because the grouping complexity of these algorithms depends on the pre-defined number of grid cells and not on the number of samples in the data. Therefore, the greatest challenge of these algorithms is to determine the best strategy for constructing the grid structure.

STING [126] is a statistical and grid-based index structure that efficiently processes region queries on databases. It creates a tree structure dividing the data space into regular cells at different levels of resolution, where each cell points to some cells of the next lower level. Some statistical information is computed and stored for each cell. A query is processed from the root until the leaves of the tree according to the likelihood of their relevance. Only children of relevant cells are recursively explored and the search ends when the lowest level of the index structure has been searched. The time complexity of a region query is $O(l)$, where l is the number of leaves of the tree. CLIQUE [5] finds dense regions (clusters) in lower-dimensional subspaces of numerical datasets as cliques in a graph.

2.5 Spectral and graph clustering

The history of *spectral clustering* (or *graph clustering*) dates back to [37] where the authors suggested that the underlying partitions in a dataset could be determined with the help of the eigenvectors of the adjacency matrix. Same as the density-based methods, these techniques do not make assumptions on the shapes of the clusters allowing them to detect non-convex clusters, such as spirals or other complex shapes. They have been successfully applied to image segmentation [111], text mining [33], speech processing [13], and general purpose methods for data analysis [34, 36].

The spectral clustering techniques represent the samples of a dataset N as nodes in a graph. The edges connecting the nodes are weighted by their pairwise similarity. According to [4], the operating mode of these methods can be generalized by the following three-step algorithm.

- **First step:** construct a similarity graph for all the data samples. Three common similarity graphs are k-nearest neighbor graph, ϵ-neighborhood graph, and fully connected graph.

- **Second step:** compute some graph Laplacian matrices and use their eigenvectors to embed the samples in a lower-dimensional space, where the underlying partitions are more evident. The main distinction here is whether to use a normalized or unnormalized graph Laplacian representation.

- **Third step:** partition the embedding space with a clustering algorithm such as k-Means.

These methods can also be explained from a simpler perspective, from the viewpoint of a graph cut. A *k-way cut* in a graph is the partitioning of the vertex set N into $C = C_1, ..., C_k$, such that $C_i \neq \emptyset$ for all i, $C_i \cap C_j = \emptyset$ for all i, j, and $N = \bigcup_i C_i$. The *cut weight* is defined as the sum of the weights of the edges across all partitions

$$weight(C) = \sum_{\forall (s,t) | s \in C_i, t \in C_j} w(s,t) \tag{2.4}$$

where $w(s,t)$ is the arc weight between the nodes s and t.

Partitioning by graph cuts usually aims to assign weights with high values for arcs within the partitions (assuming that the nodes within clusters have high similarity) and weights with lower values in their interface (assuming that nodes from different clusters have low similarity). The classic idea is to partition the nodes into two subsets such that the cut weight is minimized [130]. A problem is that this strategy often results in clusters of imbalanced sizes. Normalized Cut (NCut) is an efficient technique, with cluster size constraints, first proposed by [111]. [115] suggests the multi-class version of Normalized Cut. In [80], a Markov Random Walk view of spectral clustering is presented and the Modified Normalized Cut (MNCut) technique is proposed to handle an arbitrary number of groups.

Graph clustering is related to divisive hierarchical clustering as many methods partition the set of nodes using their pairwise similarity matrix to obtain the final groups. In [133], Zahn introduces an approach that computes a Minimum Spanning Tree (MST) in a graph and removes successively the edge with the highest inconsistency measure. One inconsistent edge is one whose weight is much higher than the average weight of the edges in its neighborhood. The authors in [16] formulate the pairwise clustering problem by relating clusters to maximal dominant sets, which are a continuous generalization of cliques in a graph.

The *Optimum-Path Forest (OPF)* framework (see Section 3.6) defines a graph topology among the samples to exploit their optimum connectivity in the feature space. In [101], the authors suggest a data clustering technique[2] based on this framework and the Mean-Shift algorithm (see Section 3.7) which has been successfully tested in some real-world

[2]This clustering technique can be seen as graph-based, density-based, and representative-based. In the last case, the representatives are the roots of a forest and the distances are optimum-path values.

applications [102, 23, 107]. This method is closely related to the present master thesis because our main goal is to extend it to support large datasets (see Chapter 4).

2.6 Clustering of large datasets

With advances in software and hardware technology, data collection has become easier in a wide variety of scenarios. A large number of clustering techniques have been developed to efficiently handle such large-size datasets. According to [59], most of them can be classified into the following five categories.

- **Efficient nearest neighbor search:** Sometimes deciding the cluster membership of each sample requires a nearest neighbor search in high-dimensional feature spaces. Algorithms that efficiently handle this type of search are either tree-based (e.g., kd-tree [85]) or random projection based (e.g., Locality Sensitive Hash [21]).

- **Data summarization:** To improve the grouping performance, these methods first summarize a large dataset into a relatively smaller one and then partition the reduced data. The samples of the original dataset receive the cluster labels that their corresponding representatives acquired after the partitioning phase. BIRCH [135] compresses a large dataset into a smaller one via a clustering feature tree (CF-tree). The nodes of this tree hold all necessary information for clustering, preventing the need to keep all data in memory. METIS [64] is a multilevel partitioning algorithm composed by three steps: the coarsening phase (the vertices are successively collapsed until the graph is small enough), the partitioning phase (the small graph is partitioned into clusters), and the uncoarsening phase (the partitioning from the second step is projected back to the original graph). In [52], the authors present a divide-and-conquer technique designed to cluster large datasets under the data stream model.

- **Distributed computing:** These methods split a clustering algorithm into a number of procedures that can be executed independently by a set of machines. Dhillon *et al.* [35] suggests a parallel implementation of k-Means based on a message passing model to cluster large datasets. Google's Map-Reduce framework [31] provides an effective method to analyze large amounts of data, especially when computing linear functions over the elements of the data streams. This framework takes care of partitioning the input data, scheduling the procedure's execution across a set of machines, handling failures, and managing all communication among the machines. In [136], the authors present a k-Means clustering algorithm on the Map-Reduce platform. BoW [49] is a distributed subspace clustering algorithm that addresses the two major bottlenecks of using serial and hard clustering techniques with the Map-Reduce framework: disk delay and network delay. DBDC [62] and ParMETIS[65] are examples of a density-based distributed algorithm and a parallel graph partitioning algorithm, respectively.

- **Incremental clustering:** These techniques, in contrast to most clustering algorithms, only allow a single pass over the data stream. In [50], Fisher proposes

a hierarchical clustering algorithm, denominated COWEB, that does a single pass through the data and arranges them into a classification tree incrementally. Bradley *et al.* [20] present a scalable clustering framework based on identifying regions of the data that are compressible (data that must be maintained in memory) from regions that are discardable. The streaming scenario is closely related with incremental clustering when real-time analysis and properly accounted changing patterns are required. In order to accomplish these goals, almost all streaming methods use a summarization technique to create intermediate representations of the data. In [3], the authors suggest a micro-clustering approach that divides the clustering process into an online component, which periodically stores detailed summary statistics, and an offline component, which operates only in these summary statistics. These components are combined with a pyramidal time frame to capture the evolving aspects of the underlying data stream. The STREAM framework, which is based on the k-Medians clustering algorithm, is presented in [89].

- **Sampling-based methods:** Techniques like CURE [53] perform a clustering over a reduced sample set of a large dataset and the result is transferred to the original data. CURE is a hierarchical clustering algorithm that finds groups of non-spherical shape by using more than one representative sample per cluster. CLARA [66] and CLARANS [87] are two classic large-scale clustering algorithms based on k-Medoids that rely on a sampling process to reduce the search space of the data. CLARA first takes a sample of all data to find the k medoids, and after that, all non-sampled data are assigned to one of the already discovered k clusters. CLARANS works with all data, but in each iteration, it checks only a subset of the cluster members to find the new medoids. Rocha *et al.* [102] propose a sampling-based clustering extension of the OPF algorithm to deal with large datasets that we call $OPF-Large-Data$ (see Section 3.7.1).

.

The divide-and-conquer clustering approach proposed in this work falls into the category of *data summarization*. The idea is (1) to divide a large dataset into smaller blocks, (2) cluster each block with the OPF algorithm to obtain the corresponding representative samples (a compressed set of the samples in the block), (3) create a new dataset with the representative samples of all the blocks, (4) cluster this reduced dataset also with the OPF algorithm, and finally, (5) propagate the cluster labels obtained in the previous step to all samples of the original dataset by means of the representative samples. This method is explained in detail in Chapter 4. However, we recommend reading the next chapter first, which presents crucial content to comprehend this thesis.

Chapter 3

Related Concepts and Methods

This chapter introduces the related concepts and methods for this thesis, including the main approach — the Optimum-Path Forest (OPF) clustering.

3.1 Digital images

A *multi-dimensional* and *multi-parametric image* I is a pair $(D, \vec{I})$, where $D \subset Z^n$ is the image domain and $\vec{I}(t) = \{I_1(t), I_2(t), ..., I_m(t)\}$ is a vectorial function that assigns m scalars (image properties) to each pixel $t \in D$. For example, $\{I_1(t), I_2(t), I_3(t)\}$ may be the red, green, and blue values of t in a color image I.

This thesis is mostly concerned with algorithms that partition an image into "relevant regions", called *image segmentation* techniques.

3.2 Image segmentation

Image segmentation can be defined as the process of identifying and separating "relevant regions" in an image. This problem represents one of the greatest challenges in Image Processing and Computer Vision, especially when the relevant regions represent objects from the real world. In this case, the problem asks for effective and efficient solutions for *object recognition* and *object delineation*. Recognition determines the approximated object location in the image, while delineation is concerned with precisely defining the spatial extent of that object. Free of fatigue, humans can outperform computers in object recognition, but the other way around is true for delineation [47]. The notion of a relevant region is highly dependent on the context and automatic segmentation often fails, except under well-controlled conditions. This explains why interactive (semi-automatic) segmentation methods usually combine the superior abilities of humans for recognition with a more precise object delineation by a computer. Figure 3.1 shows an example of interactive image segmentation. The user draws colored markers inside and outside of an object to solve recognition, while the computer delineates the object by optimal marker competition — i.e., the image is interpreted as a graph, optimum paths are computed from each marker, each pixel is conquered by the marker that offers to it the minimum-cost path, and the object is defined by the union of optimum paths from the interior marker.

(a) (b)

Figure 3.1: (a) The user draws yellow markers inside the object and black markers in the background. (b) Segmentation result from the hard constraints provided by the user.

The aforementioned example refers to semantic segmentation. Relevant regions, however, may be represented by small sets of "similar" and connected pixels, named *superpixels*, which can be delineated by clustering techniques. The main challenge here is to preserve the object borders such that each object of interest can be composed by the union of superpixels.

3.3 Superpixels

Superpixel segmentation is a convenient way to considerably reduce the number of image elements from many pixels to some regions for a more efficient image analysis. Semantic segmentation, in this case, requires the identification of the superpixels that together compose the object. Pixel similarity for superpixel definition can be measured in numerous ways, by using differences in intensity, color, texture, and even distances between pixels.

Superpixels have been successfully used in many applications: medical image segmentation [129], sky segmentation [68], motion segmentation [12], multi-class object segmentation [51, 132], object detection [112], spatio-temporal saliency detection [71], target tracking [131], and depth estimation [138]. Figure 3.2 illustrates a couple of superpixel segmentation results as obtained by a popular approach called SLIC (Simple Linear Iterative Clustering) [1]. According to [121], the desirable superpixel properties are as follows.

1. **Adherence to object boundaries:** This is a crucial property to define an object as the union of its superpixels.

2. **Connectedness and hard segmentation:** Superpixels should be connected regions without overlapping — for hard segmentation, each pixel should be assigned to a single superpixel.

3. **Compactness:** Superpixels should be constrained to have uniform size and shape.

4. **Regularity:** Regular superpixels are desirable from a topological standpoint. Therefore, the number of adjacent superpixels and the size of the boundary with each adjacent superpixel should be as uniform as possible.

Figure 3.2: Image segmentation by SLIC [1] — with 64, 256, and 1,024 superpixels (approximately). This figure was obtained from `http://ivrl.epfl.ch/research/superpixels`.

Clearly, boundary adherence is the most important property in superpixel segmentation. In [1], the authors point out that a regular lattice, like in [111], is desirable when superpixels are used as nodes of a graph. Connectedness and hard segmentation are common properties in superpixel methods, even though methods based on clustering techniques commonly require post-processing to ensure connected regions. In addition to the above properties, superpixel segmentation methods should be fast, memory efficient, and simple to use.

Most superpixel segmentation approaches adopt a clustering algorithm and/or a graph-based algorithm to address the problem in one or multiple iterations of seed estimation. Several of these methods cannot guarantee connected superpixels: SLIC (Simple Linear Interactive Clustering) [1], LSC (Linear Spectral Clustering) [24], VCells (Edge-Weighted Centroidal Voronoi Tessellations) [124], LRW (Lazy Random Walks) [109], ERS (Entropy Rate Superpixels) [70], and DBSCAN (Density-based Spatial Clustering of Applications with Noise) [110]. Connected superpixels in these methods are usually obtained by merging regions, as a post-processing step, which can reduce the number of desired superpixels.

Representative graph-based algorithms include Normalized Cuts [111], an approach based on minimum spanning tree [48], a method using optimal path via graph cuts [84], an energy minimization framework [123], and the watershed transform [17, 74]. Normalized Cuts can generate more compact and more regular superpixels; however, as shown in [1], it performs below average in boundary adherence with respect to other methods. The problem with the algorithm in [48] is exactly the opposite, the resulting superpixels

can conform to object boundaries, but they are very irregular in size and shape. The performance of the method described in [84] depends on the pre-computed boundary maps which are not guaranteed to be the best in all cases. The watershed approaches [17, 73] can easily generate irregular superpixels with reasonably good boundary recall. More recently, our group has presented a graph-based framework, named *Iterative Spanning Forest*, for superpixel segmentation, which can generate the desired number of connected superpixels with high boundary adherence [6].

Among the clustering-based algorithms, it is worth mentioning Mean-Shift [29], Quick-Shift [122], TurboPixels [69], SLIC [1], geometric flow [125], LSC [24], and DBSCAN [110]. The Mean-Shift method produces irregular and loose superpixels whereas the Quick-Shift algorithm does not allow to set the number of desired superpixels. Turbopixel-based approaches are slow and fail to provide good boundary recall for complex images. SLIC is the most commonly used superpixel method, and it was shown to perform better than many other methods [1]. It relies on a regular grid for seed sampling. Once it is chosen, the seeds are transferred to the lowest gradient position within a small neighborhood. Finally, a modified k-Means algorithm is used to cluster the remaining pixels. The k-Means algorithm searches for pixels within a $2S \times 2S$ window around each seed, where S is the grid interval. For a non-regular seed distribution, some pixels may not be reached by any seed. Indeed, this might happen from the second iteration on and this labeling inconsistency problem is only solved by post-processing. In [125], Wang *et al.* propose a geometric-flow-based method of superpixel generation. The method has high computational complexity as it involves computation of the geodesic distance and several iterations. LSC [24] and DBSCAN [110] are among the most recent approaches. LSC models the segmentation problem using Normalized Cuts, but it applies an efficient approximate solution using a weighted k-Means algorithm to generate superpixels. DBSCAN performs fast pixel grouping based on color similarity with geometric restrictions and then merges small clusters to ensure connected superpixels.

In this work, we study clustering-based algorithms which are also graph-based approaches. They can interpret image elements (pixels and superpixels) as graph nodes, build an adjacency relation between them to form the arcs of the graph and partition the image into clusters (regions that put together pixels/superpixels). The set of image pixels/superpixels for clustering is called *dataset*.

3.4 Datasets

A dataset N is a collection of samples (pixels, superpixels, or other arbitrary entities) from some specific application. Each sample $s \in N$ is represented by a feature vector $\vec{v}(s) \in \Re^m$ (e.g., a vector $\vec{v}(s) = \vec{I}(s)$ of image properties where s is a pixel/superpixel). The distance (dissimilarity) between the samples s and t in the feature space $\Re^m$ is given by a function $d(s,t)$ (e.g., $d(s,t) = \|\vec{v}(t) - \vec{v}(s)\|$).

New data acquisition technologies can provide large datasets from multiple fields, such as medical imaging, remote sensing, multimedia analysis, among others. Data has grown large at a very fast pace to support research, education, entertainment, and several

other activities. *Big data* is a fashionable concept involving large and complex datasets that cannot be handled by traditional data processing techniques. Attempts to develop effective and efficient ways of handling and analyzing big data are becoming increasingly widespread (see Section 2.6), yet they face a number of practical challenges like data storage, data analysis, and data visualization.

The graph-based techniques studied in this work are meant to deal with large datasets and they can be easily applied to find clusters in any type of dataset given some *adjacency relation* between samples.

3.5 Adjacency relations and datasets as graphs

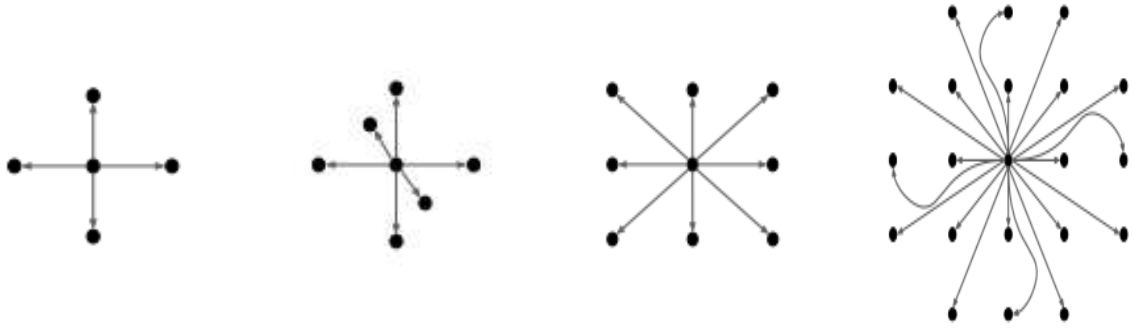
An adjacency relation A is a binary relation in $N \times N$ based on sample properties. When samples are pixels, $N \subseteq D$, the properties can be color, local texture, and pixel position. When samples are superpixels, they are usually represented by some seed pixel and the properties can be the mean color of the superpixel, seed position, color histogram of the superpixel, etc. We use $t \in A(s)$ or $(s,t) \in A$ to indicate that t is adjacent to s. Once A is established, the dataset can be interpreted as a graph (N, A) whose nodes are the samples, and arcs are the pairs $(s,t) \in A$. Examples of irreflexive adjacency relations are

$$A_1 : \{(s,t) \in N \times N | s \neq t, d(s,t) \leq r > 0\}, \tag{3.1}$$

$$A_2 : \{(s,t) \in N \times N | s \neq t, t \text{ is a } k\text{-nearest neighbor of } s \text{ in } \Re^m,\ k \geq 1\}, \tag{3.2}$$

$$A_3 : \{(s,t) \in N \times N | s \neq t, \|\vec{v}(t) - \vec{v}(s)\| \leq r_1 > 0, \|t - s\| \leq r_2 > 0\}. \tag{3.3}$$

If $N = D$ (i.e., s and t are pixels), $d(s,t) = \|t - s\|$, and $r = 1$, then A_1 is a 4-neighborhood relation and (N, A_1) is a grid graph of the image. Similarly, for $r = \sqrt{2}$, A_1 is an 8-neighborhood relation. Figure 3.3 illustrates a few examples of such spatial adjacency relations in 2D and 3D images.



(a) (b) (c) (d)

Figure 3.3: Examples of spatial adjacency relations. (a) 4-neighborhood in 2D image. (b) 6-neighborhood in 3D image. (c) 8-neighborhood in 2D image. (d) A_1 with $r = \sqrt{5}$ in 2D image.

When $d(s,t)$ is defined as a function of $\vec{v}(s)$ and $\vec{v}(t)$, the adjacency relation connects samples in the corresponding feature space $\Re^m$ (e.g., A_2 in Equation 3.2). In this case, if

s and t are pixels/superpixels, then the adjacency relation does not impose a spatial constraint. However, one can also combine image domain and feature space in the definition of an adjacency relation (e.g., A_3 in Equation 3.3). The use of A_3 for pixel clustering tends to considerably increase the number of groups as compared to A_2, but it makes possible to process the entire image domain as a dataset. The use of A_2, on the other hand, asks for a subset of samples from the image domain, called *training set*. Once clusters are computed in this training set, the cluster labels can be propagated to the remaining samples of the image domain (named in this case, *test set*). In any case, we interpret datasets as graphs and use the *Optimum-Path Forest* framework to design clustering algorithms.

3.6 Optimum-Path Forest framework

Optimum-Path Forest (OPF) is a graph-based framework that has gained considerable attention in the last years, mainly because of the promising results obtained by OPF classifiers [94, 102]. In this framework, once a dataset (training set) is interpreted as a graph (N, A), a *connectivity function* f must be provided to compute an *optimum connectivity map* $V\colon N \to \Re$. The connectivity function assigns a value $f(\pi_t)$ to any sequence of nodes $\pi_t = \langle s_1, s_2, \ldots, s_n = t\rangle$ with terminus t, such that $s_{i+1} \in A(s_i)$, $i = 1, 2, \ldots, n-1$, including the trivial case $n = 1$. The connectivity map V may result from the maximization (minimization)

$$V(t) = \max_{\forall \pi_t \in \Pi_t} \{f(\pi_t)\}, \tag{3.4}$$

where Π_t is the set of all possible paths with terminus t in the graph. The connectivity function must be defined such that *prototypes* (key samples) are identified from the maxima (minima) of a trivial connectivity map V_0 defined by $V_0(t) = f(\langle t\rangle)$. The OPF algorithm starts from $V \leftarrow V_0$ and, at each iteration, a node s, whose value $V(s)$ is optimum, offers the path extension $\pi_s \cdot \langle s, t\rangle$ to its adjacent nodes $t \in A(s)$. Whenever $f(\pi_s \cdot \langle s, t\rangle) > f(\pi_t)$, in maximization, the algorithm substitutes π_t by the extended path $\pi_s \cdot \langle s, t\rangle$, and so $V(t) \leftarrow f(\pi_s \cdot \langle s, t\rangle)$. At the end, the final connectivity map V is optimum and the graph is partitioned into an *optimum-path forest* P — i.e., an acyclic map that assigns to each node $t \in N$ its predecessor $P(t) \in N$ in the optimum path with terminus t or a marker $nil \notin N$, when $t \in S$ is a root of the map. Therefore, the final optimum path $\pi_t = P^*(t)$ can be obtained from P. The *root set* S consists of the prototypes that represent classes/clusters depending on the machine learning process: supervised, semi-supervised, or unsupervised. The prototypes may be forced by definition of f or may derive from some local property of the nodes. Examples of connectivity functions are

$$\begin{aligned} f_1(\langle t\rangle) &= \begin{cases} 0 & \text{for } t \in S, \\ +\infty & \text{otherwise,} \end{cases} \\ f_1(\pi_s \cdot \langle s, t\rangle) &= \max\{f_1(\pi_s), d(s, t)\}, \end{aligned} \tag{3.5}$$

$$\begin{aligned} f_2(\langle t\rangle) &= \rho(t) - \delta \\ f_2(\pi_s \cdot \langle s, t\rangle) &= \min\{f_2(\pi_s), \rho(t)\}, \end{aligned} \tag{3.6}$$

where $\delta > 0$, ρ is a *probability density function* (PDF), and S is a set of seed nodes (prototypes). In Equation 3.5, S can be chosen from the closest samples between categories on a complete graph for supervised learning and the OPF algorithm must minimize a *path-cost map* $V(t) = \min_{\forall \pi_t \in \Pi_t}\{f_1(\pi_t)\}$ [94]. In Equation 3.6, the root set S will be derived from the maxima of ρ where $\rho(s)$ can be estimated based on the distances between s and its k-nearest neighbors $t \in A_2(s)$. In this case, the OPF algorithm must maximize a connectivity map $V(t) = \max_{\forall \pi_t \in \Pi_t}\{f_2(\pi_t)\}$ [102]. In [102], however, the authors propose changes in the definition of A_2 and f_2 for unsupervised learning, such that a single root (i.e., cluster or optimum-path tree) will be identified for each maximum (see next Section). As presented in Equation 3.6, all nodes from a same maximum will become roots in S, over segmenting the dataset (training set).

The presented training process is accomplished by propagating the label $L(s) \leftarrow \lambda(R(s))$ for all $s \in N$, where $\lambda(R(s))$ is the class (true label) of the root node $R(s)$ in supervised learning or the cluster label of the root in unsupervised learning. The classification (class/cluster label) of a new node $t \notin N$ is performed by extending paths in P, as $P^*(s) \cdot \langle s, t \rangle$, for all $s \in N$, and propagating the label $L(t) \leftarrow L(R(s))$ of its most closest (strongest) connected root $R(s) \in S$. Consequently, class/cluster assignment is based on optimum connectivity with respect to a set S of prototypes rather than based on local distance decisions, such as in k-Means clustering, k-Nearest Neighbor classification, and several other techniques. One can derive different pattern classifiers by adapting the learning technique, the adjacency relation, the way of identifying prototypes, and the connectivity function.

Essentially, the OPF framework extends a previous approach, the *Image Foresting Transform (IFT)*, from the image domain to the feature space. The IFT is a general tool for the design, implementation, and evaluation of image processing operators based on connectivity functions [44]. The IFT reduces image processing problems to compute an optimum-path forest in a graph derived from the image. The cost of a path in this graph is determined by an application-specific function, which usually depends on local image properties along the path (such as color, gradient, and pixel position). The IFT unifies and expands many image analysis techniques, that although based on similar underlying concepts (ordered propagation, graph search, flooding, geodesic dilatation, dynamic programming, region growing, among others), are usually presented as independent methods. Those techniques can all be reduced to a partition of the image into influence zones linked to a given seed set. The influence region of each seed comprises the pixels that are "more strongly connected" to that seed than to any other, in some appropriate sense. These influence zones are the trees of the forest and each seed is the root of its corresponding tree. This idea has been used to define watershed transforms [74, 73] and to create interactive segmentation methods [98, 41, 113]. The IFT also provides a mathematically sound framework for many image processing operations that are not obviously related to image partition, such as morphological reconstruction [42], distance transforms [72], multiscale skeletons [43], multiscale fractal dimension [118], and shape saliences [119, 9, 117]. Furthermore, the IFT framework has been well succeeded in the development of image segmentation techniques based on regions [41, 74, 77, 83, 6], borders [47, 46, 45, 82], and both strategies [113, 27]. For the cases of samples as pixels and superpixels, one can say

that the OPF classifiers are IFT-based operators.

The OPF classifiers have shown advantages in some scenarios over k-Nearest Neighbors (kNN), Artificial Neural Networks (ANN), and Support Vector Machines (SVM) for supervised learning [93, 90, 94, 91, 96], and over k-Means, Mean-Shift, and Expectation Maximization (EM) for unsupervised learning [102]. They have been successfully used in several applications: rainfall occurrence estimation [95], spoken emotion recognition [58], vowel recognition [95], brain image segmentation [23, 22], active learning [105, 106], face recognition [92], petroleum well drilling monitoring [54], diagnosis of parasites [116, 107], infrared face recognition [26], background segmentation of natural images [76], among others. The OPF framework has also been extended to semi-supervised learning with classifiers outperforming many state of the art methods [8, 7]. In this work, we are mostly concerned with OPF-based data clustering.

3.7 Data clustering by Optimum-Path Forest

As proposed in [102], for a given k-nearest neighbor and arc-weighted graph (N, A_2) (see Equation 3.2 for A_2), we wish to estimate the probability density function (PDF) ρ of the samples $s \in N$ from the distances between s and its adjacent nodes $t \in A_2(s)$. The clustering we seek must be obtained by the optimum-path trees rooted at the prototype set S, derived from the maxima of ρ. Indeed, the PDF ρ is a manifold in $\Re^{m+1}$ that represents the density of the random field $\mathbf{x} = \vec{v}(s)$ as created when random choices of s lead to observations $\vec{v}(s)$ of the underlying problem. The clusters are the domes of that manifold. Therefore, one can assign $\rho(s) \leftarrow \rho(\mathbf{x})$ to all samples s that are mapped to the position $\mathbf{x}$ in $\Re^m$. This also allows to simplify the estimation of $\rho(s)$ as follows,

$$\rho(s) = \frac{1}{\sqrt{2\pi\sigma^2|A_2(s)|}} \sum_{\forall t \in A_2(s)} \exp\left(\frac{-d^2(s,t)}{2\sigma^2}\right), \tag{3.7}$$

where $|A_2(s)| = k$ and $\sigma = \max_{\forall (s,t)\in A_2} d(s,t)/3$ [1]. It can be seen that Equation 3.7 defines a *Gaussian kernel* guaranteeing that only the k-nearest samples of s are used to compute the PDF value. The traditional method to estimate a PDF is by Parzen window [38]. Equation 3.7 can provide a Parzen-window estimation when using A_1 in the feature space (see Equation 3.1). Nevertheless, this choice presents problems with differences in scale and sample concentration. Solutions to this problem lead to adaptive choices of σ depending on the region of the feature space [28]. By taking into account only the k-nearest neighbors of a sample, different concentrations are handled and the scale problem is reduced to the one of finding the best value of k within the interval $[1, k_{\max}]$, for $1 \leq k_{\max} << |N|$. In [102], the solution considers the value of k that minimizes the *normalized graph cut function*

$$C(k) = \sum_{i=1}^{c} \frac{W_i'}{W_i + W_i'}, \tag{3.8}$$

[1] This choice for σ guarantees that all adjacent samples of s in the graph are used for density computation.

$$W_i = \sum_{\forall (s,t) \in A_2 | L(s)=L(t)=i} \frac{1}{d(s,t)},$$

$$W'_i = \sum_{\forall (s,t) \in A_2 | L(s)=i, L(t)\neq i} \frac{1}{d(s,t)},$$

where $L(t)$ is the label of sample t, as assigned by clustering as the label of $R(t)$, W'_i considers all arc weights between cluster i and other clusters, and W_i considers all arc weights within cluster i for $i = 1, 2, ..., c$. Indeed, the problem of finding the best PDF is the one of finding the best value of k, and the solution requires the execution of the OPF algorithm as many times as needed, within the interval $k \in [1, k_{\max}]$, to create a candidate clustering in L and compute its normalized graph cut $C(k)$. Once the best k is found, the graph can be thought of as being node-weighted (N, A_2, ρ) (see Figure 1.1).

Adjacency A_2 (see Equation 3.2) is asymmetric and so it cannot guarantee connectivity between any pair of samples that falls on a same maximum of the PDF. In [102], the authors solve this problem by redefining the adjacency relation after PDF computation, for each evaluated value of k, in order to obtain a clustering (separation of the domes of the PDF at the valleys among them) with a single optimum-path tree (cluster) rooted at each selected maximum of the PDF. Therefore, the adjacency relation A_4 used to execute the OPF algorithm redefines A_2 as follows.

$$\begin{aligned} &\text{For all } s, t \in N, s \neq t, \text{ do} \\ &\quad \text{if } t \in A_2(s) \text{ and} \\ &\quad\quad s \notin A_2(t) \text{ and} \\ &\quad\quad \rho(s) = \rho(t), \text{ then} \\ &\quad\quad\quad A_4(t) \leftarrow A_2(t) \cup \{s\} \end{aligned} \tag{3.9}$$

Adjacency A_4 guarantees that any node selected on a maximum of the PDF will be able to reach the remaining nodes of the same maximum by an optimum path. In order to elect a single root in S per maximum of the PDF, the connectivity function must be updated as follows.

$$f_3(\langle t \rangle) = \left\{ \begin{array}{cc} \rho(t), & \text{if } t \in S \\ \rho(t) - \delta, & \text{otherwise} \end{array} \right\} \tag{3.10}$$

$$f_3(\langle \pi_s \cdot \langle s, t \rangle \rangle) = min \{ f_3(\pi_s), \rho(t) \}$$

for $\delta = min_{\forall (s,t) \in A_4 \| \rho(t) \neq \rho(s)} \| \rho(t) - \rho(s) \|$. This choice of δ preserves all maxima of the PDF. Higher values of δ work as a filter, removing clusters formed by small domes. In fact, the root set S will be represented by one node from each maximum of the optimum map V. The OPF algorithm starts by setting the trivial map $V(t) = V_0(t) \leftarrow \rho(t) - \delta$ to all nodes. During execution, each node $t \in N$ that has not been conquered by any other path than $\langle t \rangle$ (i.e., $P(t) = nil$) is the first node at a maximum of the PDF. It is then elected to be in S. The algorithm changes its trivial path value to $\rho(t)$, which makes the node to conquer the remaining ones on the same maximum and the others on the same

dome of the PDF.

Algorithm 1 presents the OPF procedure for f_3 on the graph (N, A_4, ρ). Essentially, it selects a prototype at each maximum of the PDF and then computes a path to each node t whose minimum density value along the path is maximum, among all possible paths with the same terminus t. The result is a maximum connectivity map V, a predecessor map P with one optimum-path tree (cluster) rooted at each maximum of the PDF, a root map R, and a map L with the cluster labels representing the domes of the PDF. It also creates a list O with the nodes of N ordered in a non-increasing way by the resulting connectivity values. This list is used to propagate the cluster labels to new samples. Figure 3.4 illustrates the execution of Algorithm 1 in a simple graph.

Algorithm 1 – Algorithm for Clustering by Optimum-Path Forest.

Input:	Graph (N, A_4, ρ).
Output:	Label map L, connectivity map V, predecessor map P, root map R, and list of nodes O.
Auxiliary:	Priority queue Q, variables tmp and l.

1. *Set* $l \leftarrow 1$ *and compute* δ *as described above.*
2. **For each** $s \in N$ **do**
3. $\quad$ *Set* $P(s) \leftarrow nil$, $V(s) \leftarrow \rho(s) - \delta$, $R(s) \leftarrow nil$, *insert* s *in* Q.
4. **While** Q *is not empty* **do**
5. $\quad$ *Remove from* Q *a sample* s *such that* $V(s) = \arg\max_{\forall t \in Q}\{V(t)\}$.
6. $\quad$ *Insert* s *in* O.
7. $\quad$ **If** $P(s) = nil$ **then**
8. $\quad\quad$ *Set* $L(s) \leftarrow l$, $R(s) \leftarrow s$, $V(s) \leftarrow \rho(s)$, *and* $l \leftarrow l + 1$.
9. $\quad$ **For each** $t \in A_4(s)$ *and* $V(t) < V(s)$ **do**
10. $\quad\quad$ *Set* $tmp \leftarrow \min\{V(s), \rho(t)\}$.
11. $\quad\quad$ **If** $tmp > V(t)$ **then**
12. $\quad\quad\quad$ *Set* $L(t) \leftarrow L(s)$, $R(t) \leftarrow R(s)$, $P(t) \leftarrow s$, $V(t) \leftarrow tmp$.
13. $\quad\quad\quad$ *Update position of* t *in* Q.

As discussed in [102], Algorithm 1 is more robust than Mean-Shift [25] because it does not depend on PDF gradients, it is supported by a k-nearest neighbor graph to handle different concentrations of samples, and it guarantees a single label per maximum of the PDF. On the other hand, it requires an explicit graph representation which limits the size of N and the value of k. This leads to the discussion about its extension to large datasets.

3.7.1 Extension to large datasets

Algorithm 1 takes $O(k|N| + |N|log|N|)$ operations when Q is a binary heap. The estimation of the best k requires its computation several times — e.g., if an exhaustive search is done within the interval $[1, k_{\max}]$, Algorithm 1 is executed $k_{\max}$ times. This method can become unfeasible for a large number of samples like a 2D/3D image with thousands/millions of pixels/voxels.

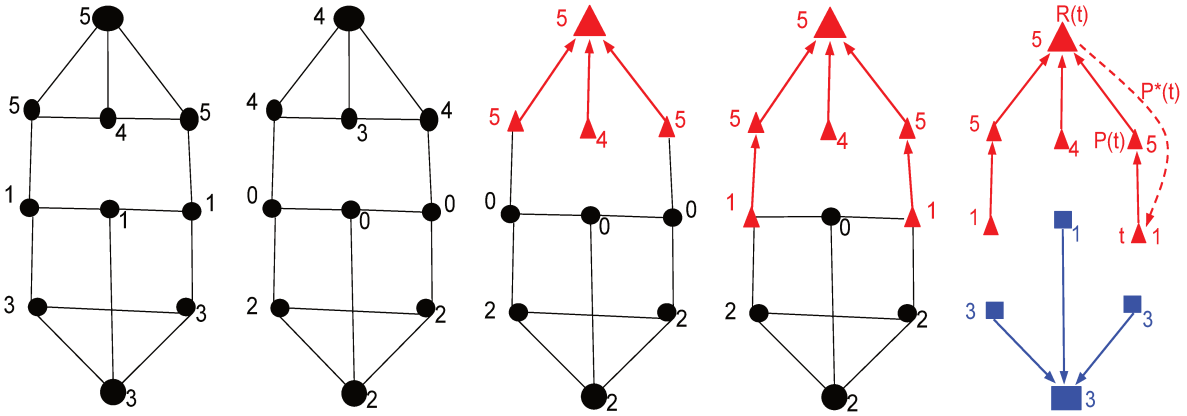


Figure 3.4: Execution of Algorithm 1 in a simple graph. (a) A 3-nearest neighbor graph whose nodes are weighted by their PDF values. There are two prototypes with values 3 and 5, as indicated by the larger dots, which are discovered later by the procedure. (b) Trivial path values after execution of Line 3 for $\delta = 1$. (c) Predecessor map (red arrows), path values (red numbers), and labels (red triangle) after the execution of the internal loop for the first node removed from Q. There is identified a prototype (largest red triangle) with PDF value equal to 5 that conquer three nodes. (d) In the next two iterations of the external loop, the non-prototype samples conquer two other nodes to be part of the cluster represented by the red triangle. (e) The optimum-path forest P, the root map R, and the connectivity map V resulting at the end of Algorithm 1. The procedure finds the other prototype of the forest (largest blue square), which in turn conquer the three remaining nodes to form a new cluster (blue square). The optimum path $P^*(t)$ (dashed line) can be recovered by following the predecessors $P(t)$ up to the root $R(t)$ for every node t.

One extension suggested in [102], with image segmentation task in mind, reduces the number of arcs in the graph by adding some spatial constraints to the adjacency computation (see A_3 in Equation 3.3). In this way, Algorithm 1 can be directly executed on the entire domain of the image/volume ($N = D$). Smaller values of r_2 in Equation 3.3 increase efficiency, but they also increase the number of clusters. Then, the choice of δ (see Equation 3.10) is very important to reduce the number of irrelevant clusters. The parameter r_1 in Equation 3.3 can be estimated as the maximum arc weight used to compute ρ (see Equation 3.7) on an uniformly sampled subset N', where $N' \subset N$ [102]. N' may consist of the resulting pixels/voxels from a 1:4, 1:8, 1:16, or other reduction factors of the image/volume resolution.

Another extension that we call $OPF - Large - Data$, also described in [102], is to execute Algorithm 1 in a randomly sampled subset N'' from the original dataset N. In this manner, N'' is known as the *training set* of the dataset N. The idea is to discover natural groups in N'', and after that, associate (or classify) each non-training sample $t \in N \setminus N''$ with one of the already discovered partitions. This technique falls within the category of *sampling-based* clustering algorithms when dealing with large datasets (see Section 2.6). In [102], the authors link each non-training sample t with the prototype that would have offered it the optimum path, if t had been part of the training forest. By considering the k-nearest neighbors of t in N'', Equation 3.7 can be used to compute $\rho(t)$, evaluate extended paths $\pi_s \cdot \langle s, t \rangle$ by f_3 (see Equation 3.10), and select the one that

maximizes the connectivity map $V(t)$ (see Equation 3.4). Let the node $s'' \in N''$ be the one that maximizes $V(t)$. Then, the classification step simply puts t in the same cluster of s'' ($L(t) \leftarrow L(s'')$). The expensive part of this process is the computation of $\rho(t)$ for all $t \in N \setminus N''$, which also requires the computation of the k-nearest neighbors of t in N''. Cappabianco *et al.* [23] considerably speeded up this *classification phase* (or *propagation phase*) by avoiding the computation of $\rho(t)$ for all non-training samples t. Essentially, their proposition is to give t the cluster label of the node with the highest path value that would have had t as a neighbor, if t had been part of the training set. Formally, [23] chooses the node s'' that satisfies

$$V(s'') = max_{\forall s \in O, d(s,t) \leq w(s)}\{V(s)\} \tag{3.11}$$

where O is the list of the graph nodes (N'') sorted in a non-increasing way by the computed path values (V), and $w(s)$ is the maximum distance between s and its k-nearest neighbors in the graph (N'', A_4, ρ). Therefore, this approach favors the nodes with the highest path values — i.e., the samples with the highest PDF values — that would have t as a k-nearest neighbor if t were in N''. This process is fast because it only needs to go through the list O until it finds the first node s'' such that $d(s'', t) \leq w(s'')$.

In spite of both extensions achieved good results in some applications, they have their limitations. The first one is used in [102] to guide the user's actions in the interactive segmentation of natural scenes. A problem is that this extension is restricted to image segmentation. Also, this technique poses a compromise between the choice of r_2 (see Equation 3.3) and the choice of δ in $f_3(\langle t \rangle)$ (see Equation 3.10) to control the number of clusters. The second extension has already been proven successful for large datasets (with about 1.5 million of voxels) when classifying gray matter, white matter, and cerebral spinal in magnetic resonance images of the brain [23]. However, in datasets with a significant imbalance among classes or datasets with many classes, the clustering result may be compromised if relevant information is lost after the sampling process that chooses the training set N''.

In this master thesis, we intend to address some of the shortcomings of these OPF-based extensions that deal with large datasets by proposing a divide-and-conquer approach. The next chapter explains in details our proposal.

Chapter 4

Divide-and-Conquer OPF Clustering

In this chapter, we present an OPF-clustering method based on the divide-and-conquer design paradigm for large datasets. The proposed method divides a dataset into parts and uses the OPF-clustering algorithm (or its variant for large datasets, *OPF-Large-Data*) to group samples in each part as well as to combine the clustering results from each part. This divide-and-conquer approach is demonstrated for two scenarios of interest: arbitrary data clustering and image segmentation. In the first scenario, it handles large datasets from arbitrary applications of data clustering and, in the second scenario, it becomes a superpixel segmentation method. We also propose improvements in the OPF-clustering algorithm.

4.1 General algorithm

Let a dataset N with $|N|$ samples, such that $|N| >> 10,000$, may be considered a large dataset irrespective of the number of features. The direct application of the OPF-clustering algorithm on N is prohibitive in processing time as well as in memory space, which is required to store a k-nearest neighbor graph with possibly $k = 500$ (see Section 3.7). Our strategy is to divide N into smaller subsets, find groups in these subsets, and combine the groups from all subsets to obtain the final partition. We are not the first to suggest this model for data clustering. Indeed, Jain *et al* [61] propose it as a possible variant when the entire dataset cannot be accommodated in the main memory.

Algorithm 2 describes our technique. We call it *OPF-Blocks-2* in reference to the fact that it only has two clustering levels. At the first level, the large number of samples in N is divided into b disjoint blocks (Line 1), so that each segment of the data (block) consists of approximately N/b samples. This number must be reasonable and sufficient to represent the natural groups in N. Otherwise, this choice will affect the performance of the method. This data division can be random or based on some application-specific strategy. Then, Algorithm 1 is used to group the samples in each block. This phase is easy to parallelize because the blocks are clustered separately. Let us assume that block i produces c_i clusters, for $i = 1, 2, \ldots, b$, or what is the same, block i can be summarized by c_i prototypes (cluster representatives) [1]. Subsequently, all the $\sum_{i=1}^{b} c_i$ prototypes

[1]Remember that in OPF clustering the roots of the forest summarize their corresponding trees (clus-

are taken as samples of a new dataset M (Line 5) to be grouped in the second level also by Algorithm 1 (Line 6). It is expected that this last result will reveal the natural number c of groups in the original dataset. Our assumption is that the samples in M summarize the original data, therefore, a clustering over these samples will represent a good approximation of the underlying partition of the dataset N. Finally, the group labels acquired in M are transferred to the original dataset N as follows. Line 8 copies the group labels of the samples in M to the same samples in N — i.e., the roots of optimum-path trees in the first level — and then to the samples of their optimum-path trees (Lines 11 and 12). Algorithm 2 returns the label map L, the root map R, and the predecessor map P (optimum-path forest) from N (Lines 8 and 12).

Algorithm 2 – OPF-Blocks-2

INPUT: Large dataset N, adjacency relation A_4, probability density function ρ, and number of parts b.
OUTPUT: Label map L, predecessor map P, and root map R.

1. *Divide N into b disjoint sets N_1,...,N_b.*
2. *Create empty set M.*
3. ***For each*** $i \in (1..b)$ ***do***
4. $\quad$ $(L_i, P_i, R_i) \leftarrow$ *Execute Algorithm 1 in* (N_i, A_4, ρ).
5. $\quad$ *Add the representative samples of N_i to M.*
6. $(L_m, P_m, R_m) \leftarrow$ *Execute Algorithm 1 in* (M, A_4, ρ).
7. ***For each*** $s \in M$ ***do***
8. $\quad$ *Set $L(s) \leftarrow L_m(s)$, $R(s) \leftarrow R_m(s)$, and $P(s) \leftarrow P_m(s)$.*
9. ***For each*** $i \in (1..b)$ ***do***
10. $\quad$ ***For each*** $s \in N_i \setminus M$ ***do***
11. $\quad\quad$ *Set $u \leftarrow R_i(s)$.*
12. $\quad\quad$ *Set $L(s) \leftarrow L(u)$, $R(s) \leftarrow R(u)$, $P(s) \leftarrow P_i(s)$.*

The time complexity of Algorithm 1 is $O(k|N| + |N| \log |N|)$, as previously mentioned. By assuming $k \gg \log |N|$ and Algorithm 1 is executed $k_{\max}$ times to discover the best value of k, we may conclude that the OPF-clustering technique runs in $O(k_{\max} * k * |N|)$. Algorithm 2 depends on Algorithm 1 to group samples in each subset of N as well as to find groups in M. Assuming that M has a similar number of samples to that of the blocks in the first level (i.e., $|N|/b$ samples), we may conclude that Algorithm 2 executes in $O((b+1) * k'_{\max} * k' * |N|/b) = O(k'_{\max} * k' * |N|)$ for $k'_{\max} < k_{\max}$ and $k' < k$. This makes the proposed method not only viable for large datasets but also more efficient than Algorithm 1, especially for smaller values of $k'_{\max}$ and k', which are obtained as b increases.

Algorithm 2 can be easily extended to a higher number of levels than two, but we found two levels enough for the datasets used in this work. When the number of samples in a dataset is considered very large (for instance in an image, where $|N| >> 200,000$), we prefer to partition the data into a smaller number of blocks and use the variant *OPF-Large-Data* to cluster each block rather than partition them into a larger number of

ters).

blocks. In this case, Algorithm 1 is executed in reduced training sets and the resulting cluster labels are propagated to the remaining samples of the corresponding blocks (see Section 3.7.1). By that, our algorithm makes possible to use the method of Rocha *et al.* [102] with considerably larger training sets. For instance, if *OPF-Large-Data* can cluster a very large dataset within a reasonable time by using x training samples, we can affirm that our technique can cluster the same dataset in comparable time by using xb training samples, where b is the number of blocks. This flexibility is important for some large datasets, where reduced numbers of training samples can compromise the clustering results due to the lack of data information.

We can use Algorithm 2 to cluster any large set of arbitrary data. The dataset needs to be divided by random sampling in the absence of domain information on the samples. Ideally, each block should have a good representation of the underlying partition, but not an excessive number of samples that can compromise the efficiency of the technique. In fact, this number also depends on the dimensionality of the data. The summarized dataset in the second level must also have a balanced size. If this dataset is formed by many samples, it means that the samples were not sufficiently grouped in the first level and the results of the proposed technique will be quite similar to those of the Algorithm 1 — i.e., the second level of the proposed method would constitute the only level of Algorithm 1. On the contrary, if this number is small, important grouping information may be lost and the result may be poor. It is valid to clarify that these are mostly assumptions. There is no one truth when it comes to obtaining the best parameters (number of blocks, number of samples per block, size of the summarized dataset) and the technique must be rigorously tested on each dataset in which it is used. Algorithm 2 is depicted in Figure 1.2 when clustering a toy dataset.

In the next section, we propose improvements in the original OPF-clustering technique, especially in the processing time to estimate the best value of k.

4.2 Improving the estimation of the k-nn graph

Figure 4.1 illustrates the pipeline of the original OPF-clustering method, which includes several executions of the OPF algorithm (Algorithm 1) to estimate the best value of k (i.e., the most suitable k-nn graph for the problem). For a given value of $k_{\max} \ll |N|$, the best value of k must be estimated within $[1, k_{\max}]$. In order to avoid the computation of $k_{\max}$ k-nn graphs, which takes $O(|N|^2)$ each, the traditional approach starts by building a $k_{\max}$-nn graph with the $k_{\max}$ nearest neighbors (adjacent nodes) of each node sorted, such that the $k_{\max}$-nn graph actually includes every k-nn graph for $k \in [1, k_{\max}]$. First, we parallelized the construction of the $k_{\max}$-nn graph. The identification of the k-nearest neighbors could also be based on a k-d tree or other specialized structures, whose complexity time is $O(|N| \log |N|)$, but its difficulty for parallelization is considerably higher.

The authors in [102] find the best value of k by exhaustive search within $[1, k_{\max}]$. The normalized cut in the k-nn graph is used as the criterion for minimization, as computed from the group labels propagated by the OPF algorithm. As drawback, for each candidate k, the technique must compute the PDF of all nodes (light green component), execute

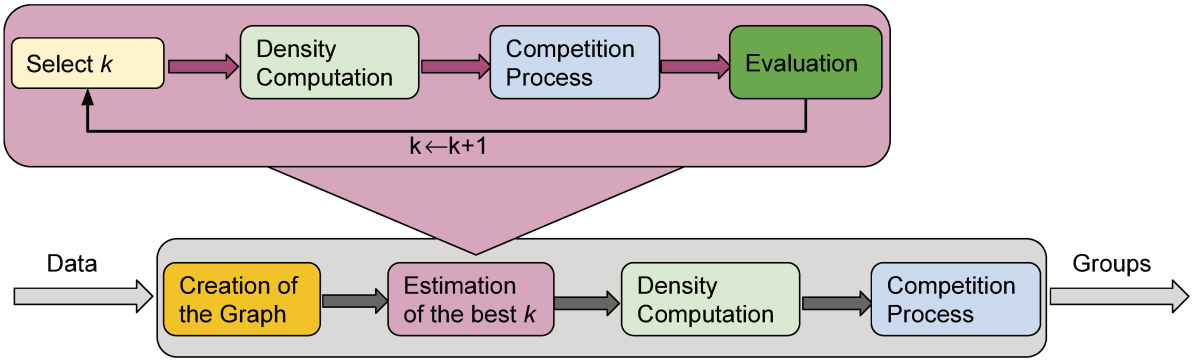


Figure 4.1: Workflow of the OPF-clustering technique as proposed by Rocha *et al.* [102].

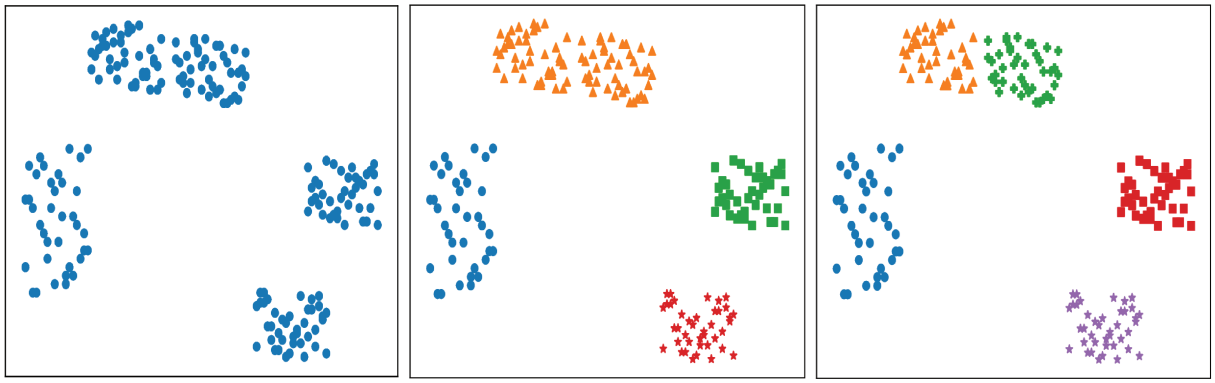
Algorithm 1 (light blue component), and evaluate the normalized cut (see Equation 3.8) produced by the group labels on the graph (green component). We have parallelized the PDF computation and the clustering evaluation. However, the parallelization of Algorithm 1 is more difficult due to its priority queue. We then propose here a heuristic to reduce the number of iterations of this pipeline (rose pipeline).

The parameter k represents the observation scale of the data in the feature space. Depending on its value, Algorithm 1 can output more or less groups (see Figure 4.2). Basically, higher values of k may produce a single cluster while lower values of k separate the samples in more groups. A problem with this technique is that it cannot partition the data into a specified number of clusters without playing with the parameter $k_{\max}$. In [2], the authors solve the issue by performing the clustering at different levels of abstractions (scales) to be as close as possible to the number of clusters required by the application. In our case, we are not interested in establishing the resulting number of groups to the method, but in reducing the computational time of the search of k within $[1, k_{\max}]$. Therefore, our idea is to start the search at $k = k_{\max}$ and stop it whenever the first local minimum of the normalized cut function is found (see Figure 4.3). By that, we are choosing fewer clusters but better estimating the PDF based on higher values of k. It turns out that this heuristic usually produces good results with earlier search termination. In Section 5.1.6, we compare the results obtained by *OPF-Large-Data* based on the exhaustive search and our heuristic search. Of course, higher is $k_{\max}$ more likely is that the exhaustive and heuristic searches will choose different values of k, being the chosen value in the former lower than the value selected by the latter. However, one can always reduce the upper limit $k_{\max}$ in order to make them equivalent again. Another possible solution to speed up the estimation of k is through meta-heuristic optimization [30].

The next section discusses the application of the proposed technique to image segmentation.

4.3 Algorithm for image segmentation

When a dataset consists of image pixels, the expected clustering result is a partition of the image into regions, called superpixels, such that the image objects can be represented by the union of their superpixels. We have addressed this problem by divide-and-conquer



(a) (b) (c)

Figure 4.2: 2D projections of a toy dataset by the t-SNE algorithm. The number of groups reduces as the scale k increases. Higher values of k produce (a) a single group and smaller values of k produce (b) four and (c) five groups, for instance.

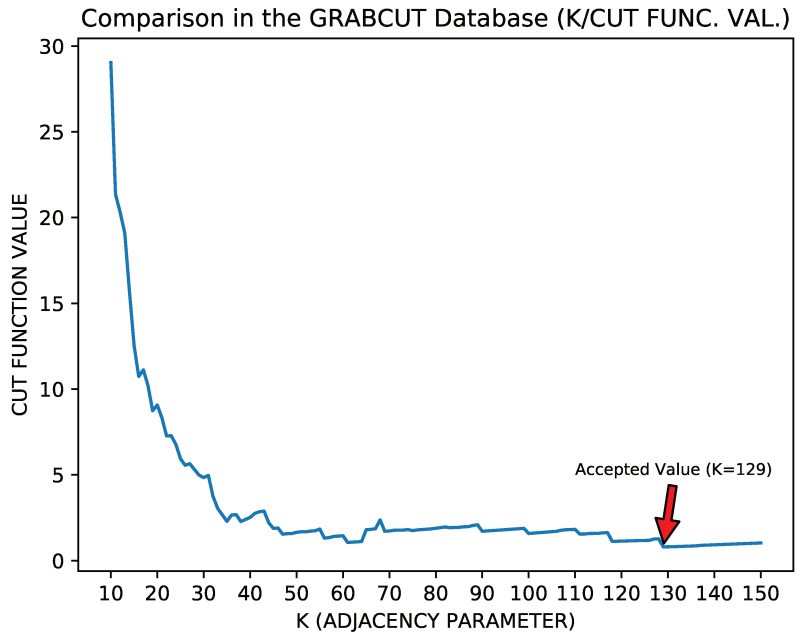


Figure 4.3: Normalized cut function (Equation 3.8) evaluated on the results produced by *OPF-Large-Data* in an image of the GrabCut database, when varying $k \in [1, k_{\max}]$ for $k_{\max} = 150$. In this case, both the exhaustive and heuristic searches detect the same optimum value $k = 129$, but the latter takes 127 iterations less than the former.

OPF clustering as follows.

We first divide the image into a grid of blocks in order to take advantage of their spatial information. One question is how to choose the training samples in each block. The number of blocks and their corresponding training set sizes are also important issues. These three subjects are addressed in Section 5.1.6. Many blocks or larger training sets seems to allow for better results because of the usage of more training samples; however, these decisions slow down the run-time of the technique and may be unnecessary. Therefore, as we will see in the next chapter, it is enough to work with few blocks and reduced training

sets per block, as obtained by grid sampling in each block. Algorithm 1 is then applied to the reduced training set of each block and the cluster labels are subsequently propagated to the remaining samples in the block— i.e., we preferred to use the variant named *OPF-Large-Data*. We parallelized this label propagation operation. Afterward, it is possible to merge adjacent clusters from neighboring blocks by means of a post-processing, a variant of the algorithm named *OPF-Blocks-1*, or to continue the process by clustering the roots from each block in a second level and then propagating the cluster labels through them to all image pixels (*OPF-Blocks-2*). Algorithm 3 shows this two-level version for image segmentation.

Algorithm 3 – IMAGE SEGMENTATION BY OPF-BLOCKS-2

INPUT: Dataset corresponding to an image N, adjacency relation A_4, probability density function ρ, and number of blocks b.
OUTPUT: Label map L, predecessor map P, and root map R.

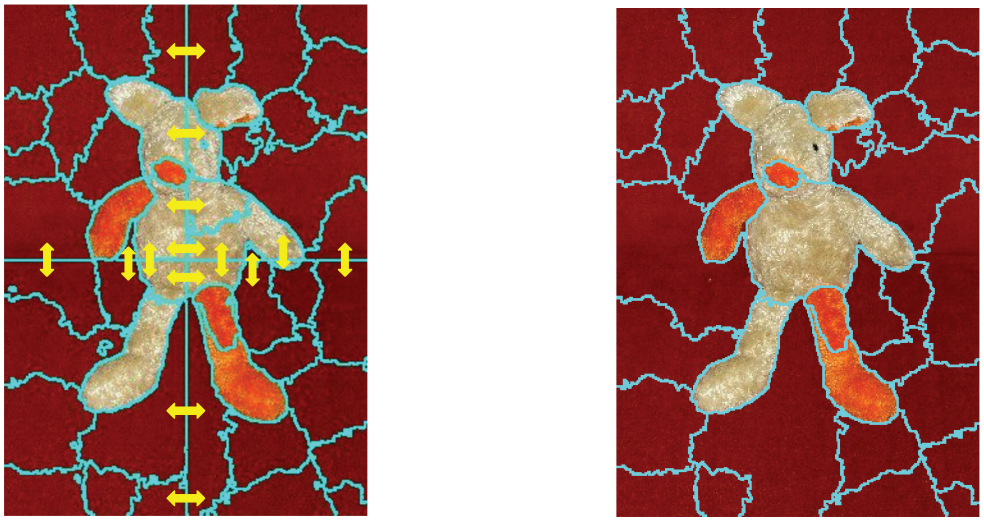
1. *Divide N into b disjoint and compact blocks N_1,...,N_b.*
2. *Create empty set M.*
3. ***For each*** $i \in (1..b)$ ***do***
4. *Select a reduced training set T_i from N_i by grid sampling.*
5. $(L_i, P_i, R_i) \leftarrow$ *Execute Algorithm 1 in* (T_i, A_4, ρ).
6. ***For each*** $s \in N_i \setminus T_i$ ***do***
7. *Set $u \leftarrow$ the node returned by Equation 3.11.*
8. *Set* $L_i(s) \leftarrow L_i(u)$, $R_i(s) \leftarrow R_i(u)$, *and* $P_i(s) \leftarrow u$.
9. *Add the representative samples of T_i to M.*
10. $(L_m, P_m, R_m) \leftarrow$ *Execute Algorithm 1 in* (M, A_4, ρ).
11. ***For each*** $s \in M$ ***do***
12. *Set* $L(s) \leftarrow L_m(s)$, $R(s) \leftarrow R_m(s)$, *and* $P(s) \leftarrow P_m(s)$.
13. ***For each*** $i \in (1..b)$ ***do***
14. ***For each*** $s \in N_i \setminus M$ ***do***
15. *Set $u \leftarrow R_i(s)$.*
16. *Set* $L(s) \leftarrow L(u)$, $R(s) \leftarrow R(u)$, $P(s) \leftarrow P_i(s)$.

While *OPF-Blocks-1* is limited to merge adjacent clusters from neighboring blocks, *OPF-Blocks-2* can merge clusters from blocks in any part of the image. The merging procedure in *OPF-Blocks-1* consists of computing a color histogram for the superpixels and join adjacent pairs whose the Bhattacharyya coefficient between them is close to 1. The Bhattacharyya coefficient (BC) is defined as

$$BC = \sum_{x \in X} \sqrt{p(x)q(x)} \tag{4.1}$$

where p and q are discrete probability distributions over the same domain X. Figure 4.4 depicts this procedure.

To produce the final segmentation, we need to apply a relabelling on the clustering result of the methods because the obtained clusters are not restricted to be compacted while the superpixels are. In fact, this is not the only post-processing that we recommend.



(a) (b)

Figure 4.4: Image segmentation by *OPF-Blocks-1* with four blocks. (a) After clustering the pixels by *OPF-Large-Data* in each block. Adjacent superpixels from neighboring blocks are marked by yellow arrows. (b) After adjaceny superpixel merging by the post-processing, the block boundaries are removed, making the segmentation more natural.

We also propose to apply a smoothing to the resulting superpixels and a filtering to remove noise (small sets of pixels). Both operations are implemented using the Image Foresting Transform framework (see Section 3.6). Figure 4.5 depicts all the pipeline of *OPF-Blocks-2* for image segmentation.

In this chapter, we discussed in detail our divide-and-conquer proposal and how it fits when clustering large arbitrary datasets and segmenting images. In the next chapter, we evaluate and compare this technique with other relevant methods in the two scenarios of interest.

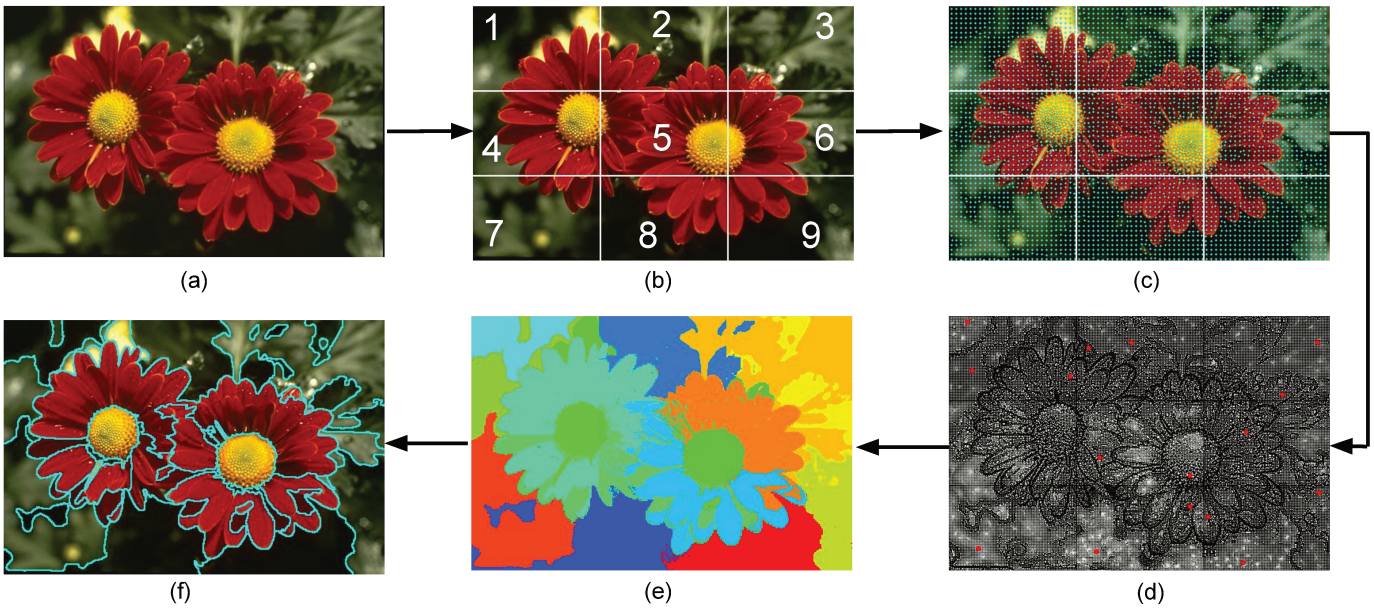


Figure 4.5: Pipeline of *OPF-Blocks-2* for image segmentation. (a) Input image. (b) The image is divided into 9 blocks. (c) There is selected a training set (blue points) by grid sampling in each block. (d) The PDF values of the samples are estimated separately in each block. Brighter values indicate samples with higher values of the PDF. Afterward, each block is clustered with OPF-Large-Data, the prototypes of the groups are promoted to the second level where they are clustered with the OPF algorithm. The roots of the final forest are indicated with red points. (e) The group labels obtained in the second level are propagated to all samples of the image. Each color indicates a different group. (f) Resulting superpixels after relabelling, smoothing, and filtering operations.

Chapter 5

Experimental Results

This chapter describes the experiments and results of the proposed algorithms for two scenarios: image segmentation and arbitrary data clustering. For image segmentation, the samples are the pixels of a given image (dataset) as represented by their color and spatial location. For data clustering, the datasets have different numbers of features, classes, and samples per class, representing distinct applications. For each scenario, we use common datasets to compare methods according to popular metrics of effectiveness. A discussion about the experimental results is presented at the end of the chapter.

5.1 Image segmentation

Traditional clustering techniques, or methods based on clustering, have been used to group pixels based on their color similarity and difference in location on the image. Next, we compare two clustering techniques derived from our proposal, OPF-Blocks-1 and OPF-Blocks-2, against some popular superpixel generation methods and other clustering algorithms. To evaluate the methods across different application domains, we select databases involving natural, biological, and medical images.

5.1.1 Image databases

We use three databases of 2D images with their corresponding ground-truth segmentation to measure the effectiveness of the methods to adhere to the boundaries of a given object of interest.

The first database corresponds to 50 natural and colorful images from the GrabCut database [103]. The image size varies from 339.1 kB (113,032 pixels) to 921.6 kB (307,200 pixels). Some examples of images are shown in Figure 5.1.

The second database is formed by 36 color images of different parasites where some are connected to impurities. The goal in these images is to isolate the pixels that represent the parasites. However, impurities may overlap the parasites and/or present similar sizes, shapes, colors, and textures. The image size varies from 391.9 kB (97,280 pixels) to 2.1 MB (698,880 pixels). Some examples of images of this database can be observed in Figure 5.2.

Figure 5.1: Examples of images in the GrabCut database.

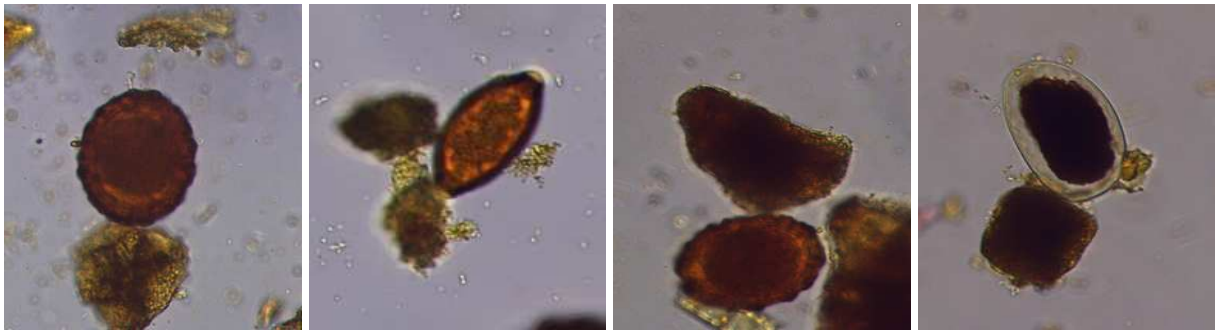


Figure 5.2: Examples of images in the Parasites/Impurities database.

The third database contains 29 images obtained from slices of 10 thoracic computed tomography (CT) studies. The object of interest is the liver in each slice. The images are gray-scale and their sizes vary from 762 kB to 805.4 kB, all having 262,144 pixels. Figure 5.3 exhibits some images of this database.

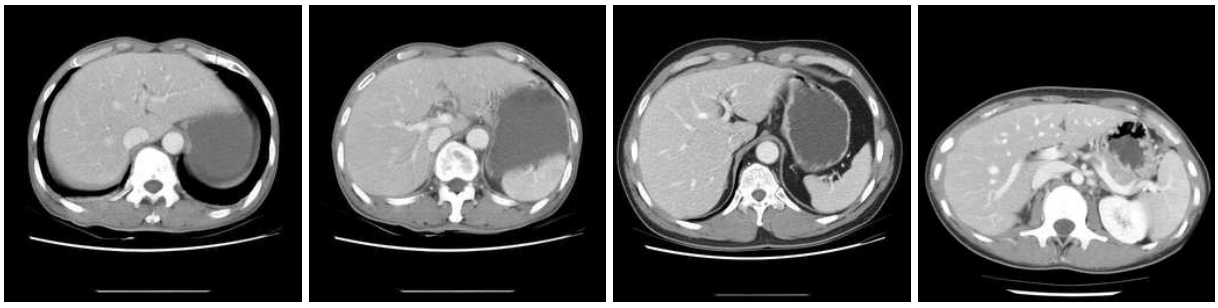


Figure 5.3: Examples of images in the Liver database.

5.1.2 Compared methods

OPF-Blocks-1 and OPF-Blocks-2 are compared to two state-of-the-art superpixel generation methods (SLIC and Quick-Shift), one method based on the watershed transform, and two generic clustering algorithms (OPF-Large-Data and k-Means). In particular, the comparison between OPF-Large-Data (see Section 3.7.1) and the divide-and-conquer methods is interesting. Currently, OPF-Large-Data is the principal option when trying to cluster large datasets with the OPF framework, and in this work, we are assuming that our technique improves or at least obtains similar results to the previously developed OPF-based extensions. OPF-Large-Data is always executed with 1500 training samples

when clustering each image or block in the divide-and-conquer extensions. OPF-Blocks-1 and OPF-Blocks-2 divide the images into 4, 9, or 16 blocks at the first level, so in turn they are using a training set with size between 4 * 1500 and 16 * 1500 samples.

Some superpixel generation methods require a post-processing to ensure that the generated superpixels are connected, for others, the algorithm itself ensures this connectivity. SLIC, k-Means, and the OPF-based extensions all require this procedure, while the watershed-based method does not require it. This is not the only post-processing that we apply in the experiments. We also remove small superpixels (noise) from the result of each compared method, and in some cases, we apply image smoothing based on diffusion filtering [1]. We define the descriptor for these databases as explained in Section 5.1.4. Below we give a brief description of each compared method.

SLIC: SLIC is a linear (in the number of pixels) algorithm, and it is by far the most commonly used superpixel method. In Section 3.3, there is a short description of this technique. SLIC allows to specify the number of desired superpixels and to regulate the compacity of them. The superpixels for our experiments are generated using the implementation provided in the authors' webpage[2].

Quick-Shift: Quick-shift uses a mode-seeking segmentation scheme [122]. It initializes the segmentation using a medoid-shift procedure. Then, it moves each point in the feature space to the nearest neighbor that increases the Parzen density estimate. Quick-shift is a rather slow algorithm, with a $O(dN^2)$ complexity where d is a small constant [122]. Also, it does not provide user control over the size or number of superpixels. Previous works have used Quick-Shift for object localization [51] and motion segmentation [12]. The superpixels for our experiments are generated using publicly available source code[3].

Watershed: The watershed-based approaches perform a gradient ascent starting from local minima to produce watershed lines that separate catchment basins. The resulting superpixels are often highly irregular in size and shape and do not exhibit good boundary adherence for small numbers of superpixels. We use the algorithm explained in [74] to generate the superpixels.

k-Means: We utilize a basic implementation of k-Means to cluster the pixels of the images. The complexity of this algorithm is $O(KNI)$ where K is the desired number of groups and I the maximum number of iterations until convergence. The initial cluster centers are chosen with grid sampling.

OPF-Large-Data: We use the OPF clustering technique applied to image segmentation as described in the last extension of Section 3.7.1. The training samples for each dataset (image) are chosen with grid sampling.

OPF-Blocks-1: This is the approach described in Algorithm 3 but with only one level. The training set in each block is formed with grid sampling. It uses the suggested local search to find the adjacency parameter k (see Section 4.2) when clustering each block and the merging post-processing explained in Section 4.3.

OPF-Blocks-2: This is the proposed technique, described in Algorithm 3, with the

[1] The diffusion filtering is maintained only if it improves the segmentation result.

[2] `http://ivrl.epfl.ch/supplementary_material/RK_SLICSuperpixels/`

[3] `http://www.vlfeat.org/download.html`

two levels. As in OPF-Blocks-1, for each block, the training samples are chosen with grid sampling and the suggested local search is used to find the adjacency parameter k.

5.1.3 Evaluation metrics

We employ two widely used boundary adherence measures for evaluating the quality of superpixels. The first one is the *boundary recall (BR)* which measures the fraction of the ground-truth boundaries overlapping the segmentation boundaries in an image within a certain tolerance distance d of pixels. We use $d = 2$ for our experiments[4]. The second used metric is the *under-segmentation error (UE)* which does not penalize over-segmentation and indicates how well the superpixels adhere to the object boundaries. There are different definitions for this metric [1, 86], so we utilize the free parameter definition presented in [86]. Every ground-truth segment G_i having an overlap with a superpixel S_j divides it in *in* and *out* parts, denoted by S_j^{in} and S_j^{out}, respectively. The first part represents the set of pixels in $S_j \cap G_i$, and the last one comprises the set of pixels in S_j that lies outside $S_j \cap G_i$. UE represents the smallest error introduced by either adding S_j^{out} to the segment or by omitting S_j^{in}. Let M be the number of ground-truth regions, I_s a superpixel segmentation, and N the number of pixels in the image. UE is defined as

$$UE(I_s) = \frac{1}{N}\left(\sum_{i=1}^{M}\left(\sum_{S_j|S_j\cup G_i\neq\emptyset} \min(|S_j^{in}|, |S_j^{out}|)\right)\right) \tag{5.1}$$

where $|.|$ denotes the size of a set of pixels.

In addition to computing the boundary recall and the under-segmentation error, we also measure the mixture between background and object (the object of interest) in the resulting clusters. In OPF-based methods, it is assumed that all samples from an optimum-path tree have the same label of their root. Therefore, we assign the correct label to each root of the forest and propagate them to the remaining samples of the corresponding optimum-path trees. In this way, the *purity* or *accuracy* of the clustering is defined as the percentage of correct classifications obtained by the previous procedure. For the other methods which are not based on the optimum-path forest but have a clear representative for each cluster — e.g., in k-Means the representatives are the closest samples to the centers of the clusters —, we use the same idea to compute the accuracy of the clustering. We assign the true label to each cluster representative and propagate it to the rest of the group. We also calculate the F_1 *score* or *Dice similarity coefficient (DSC)* from the label propagation result. The Dice coefficient is defined as

$$DSC = \frac{2 * TP}{2 * TP + FP + FN} \tag{5.2}$$

where TP is the number of true positive samples, FP the number of false positive samples, and FN the number of false negative samples. In the images, we assume that the positive samples are the samples in the object of interest and the negative ones are the samples

[4]This tolerance value is usually adopted to cope with human errors when generating ground-truth segmentations.

in the background.

We carry out the label propagation experiments in all three databases for SLIC, k-Means, and the OPF-based extensions. To compute the Dice metric, we do some post-processing to the images resulting from the label propagation phase. We apply morphological open and close operations to reduce noise, and we only remain with the largest object component because we know that the evaluated images have a single object of interest.

5.1.4 Defining a simple and effective descriptor for the samples in the evaluated images

To find a simple and good enough descriptor for the samples in the tested databases, we help ourselves with the t-Distributed Stochastic Neighbor Embedding (t-SNE) data visualization technique [75]. It is clear that we need the color information for pixel clustering, but it is not so obvious if we also need the spatial information or color information from a neighborhood close to the pixel. We choose the CIELAB color space to encode color information because of its common use in superpixel methods.

Figure 5.4 shows a natural image of the GrabCut database and the 2D projections of different feature vectors (descriptors) with the t-SNE technique. It is easy to realize that the spatial information is important for superpixel segmentation (Figure 5.4c) because pixels of background and object can have similar color in the same image (Figure 5.4a). Not by choice SLIC [1] defines a distance involving both color and spatial differences. Also, we can see from Figure 5.4d that adding the color data from a pixel's neighborhood, without adding together its spatial information, decreases the spatial information relevance and worsens the separability between object and background in the feature space. Therefore, we define the descriptor for the samples of colorful images with both, the color and spatial information of the pixel[5].

For the gray-scale images, corresponding to the thoracic computed tomography slices, we determine a feature vector formed by the brightness value and the spatial information of the pixel, in addition to the brightness data of the pixels in the 8-neighborhood. This descriptor results in a clear separation between liver and not liver samples in the t-SNE projection.

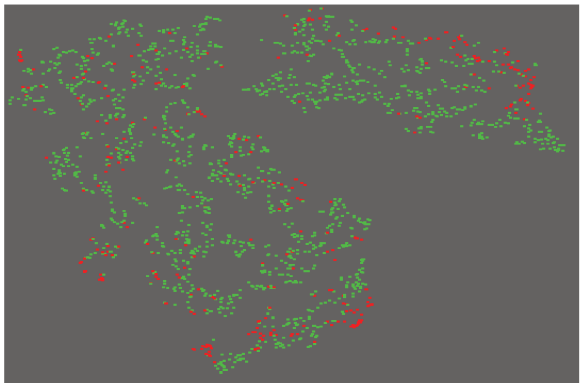
5.1.5 Experimental methodology

All experiments are executed on a server with a processor Intel Core i7-3770K CPU @ 3.50GHz x 8 and a memory RAM of 32GB. We randomly divide each database into two sets: a *training set* and a *test set*. The images in the training set are used to tune up the hyper-parameters of the compared methods[6]. The best hyper-parameters are found by grid search. All the results shown below correspond to the execution of the methods

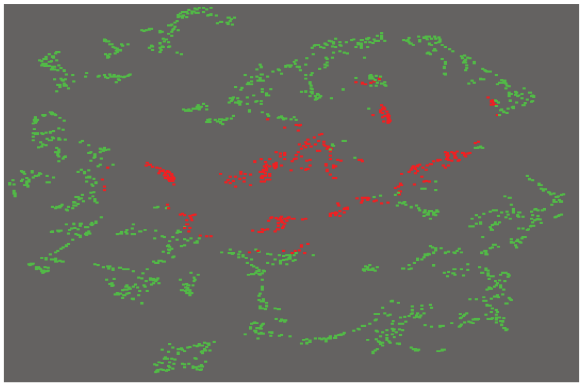
[5]We could also have tried more complex features as texture descriptors, but the idea was to create a simple feature vector.

[6]Do not confuse the training set for a database of images (it is formed by images to tweak the hyper-parameters of the compared methods) with the training set of a large dataset according to the OPF-Large-Data clustering technique (it is formed by samples or pixels in this case).

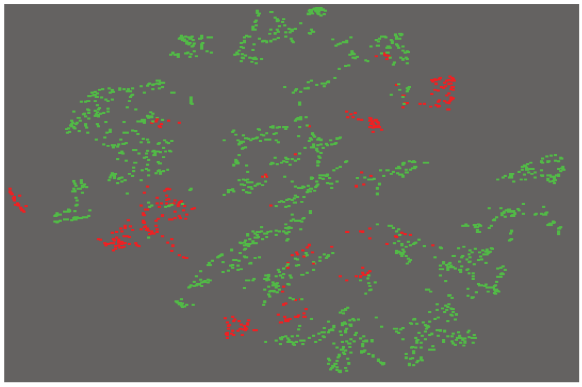
(a) A natural image from the GrabCut database.



(b) 2D projection of the descriptor formed by only the CIELAB color information of the pixel.



(c) 2D projection of the descriptor formed by both the CIELAB color and spatial information of the pixel.



(d) 2D projection of the descriptor formed by both the CIELAB color and spatial information of the pixel, in addition to the CIELAB color data of the pixels in the 4-neighborhood.

Figure 5.4: 2D projections of different descriptors for a natural image of the GrabCut database with the t-SNE technique.

exclusively on the images of the test set. We average the metrics obtained in these images to create the comparative graphics. The test set is greater than the training set in the three databases. The databases are divided as follows:

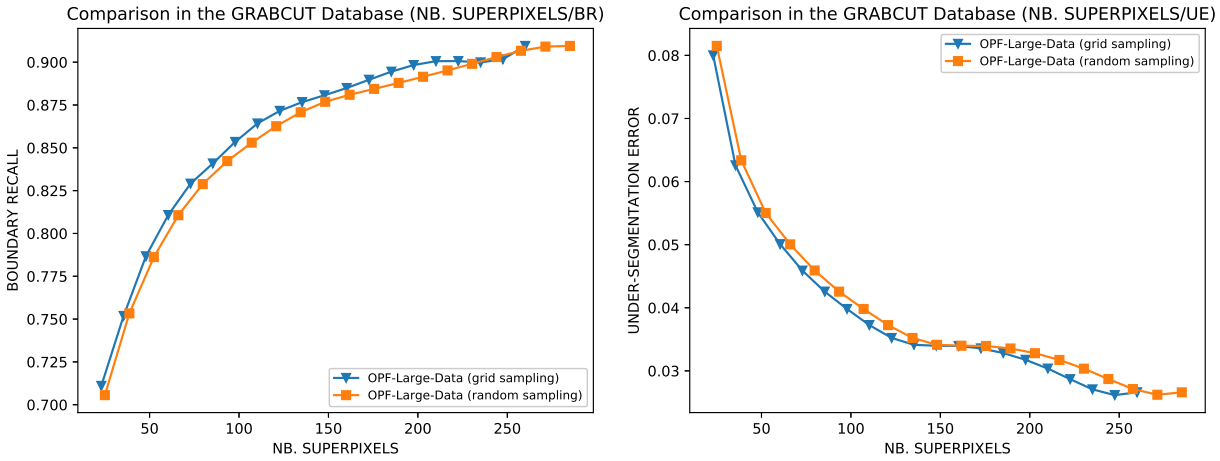
- **GrabCut database: 15** images for training and **35** images for testing.
- **Parasites/Impurites database: 12** images for training and **24** images for testing.
- **Liver database: 11** images for training and **18** images for testing.

5.1.6 Results

In this section, we show the results of the experiments performed. First, we answer some questions related to the OPF-clustering technique and our proposal that are presented in Section 4.2, and finally, we compare our technique with the other methods in the selected image databases.

Is there any difference between random sampling and grid sampling when forming the training set of OPF-Large-Data?

Figure 5.5 compares the segmentation results of OPF-Large-Data, according to two different strategies used when creating a training set of 1500 samples, in the GrabCut database. It can be observed that creating the training set with grid sampling is a little better than forming it with random sampling. This result is easy to understand. A randomly formed training set with not enough number of samples can omit some components of the image. With grid sampling, lose relevant information is harder because the training samples are uniformly distributed in the data space. For example, it can be seen in Figure 5.6a how some borders of the object are lost (marked with yellow arrows) after clustering the image by OPF-Large-Data using a training set formed with random sampling. We cannot say the same for a training set created with grid sampling according to Figure 5.6b.



(a) Comparison according to the boundary recall metric.

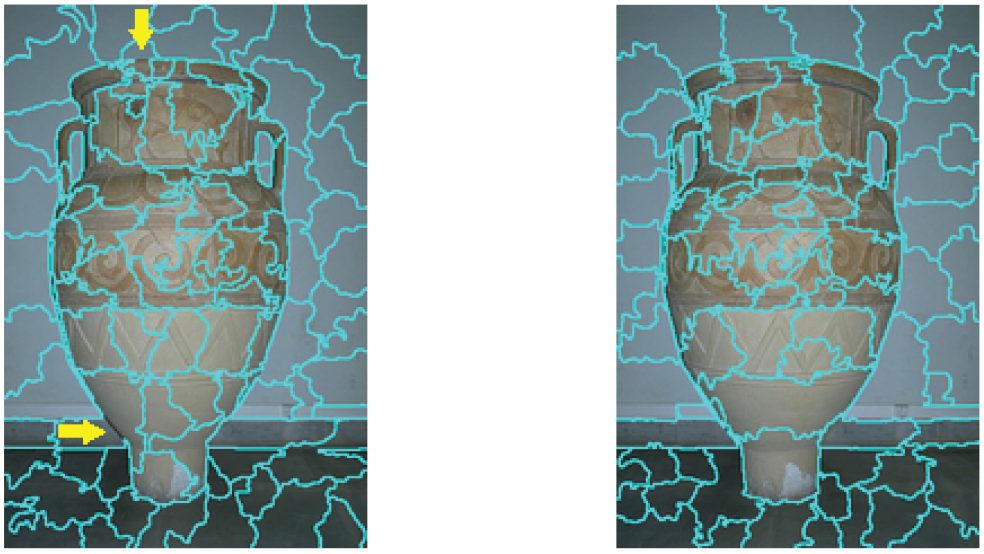
(b) Comparison according to the under-segmentation error.

Figure 5.5: Comparisons between the segmentation results of OPF-Large-Data, in the Grabcut database, when different strategies to form the training set of the method are used.

In all the image segmentation experiments shown from now on, the compared techniques that need to do a sampling — e.g., when selecting the k initial centers in k-Means, when forming the training set in OPF-Large-Data, and when selecting the training samples in the blocks of OPF-Blocks1 and OPF-Blocks-2 —, perform a grid sampling and not a random sampling.

Does a larger number of training samples in OPF-Large-Data imply better results?

Figure 5.7 shows comparisons between the segmentation results obtained by OPF-Large-Data in the images of the Grabcut database when different training set sizes are used. It is clear that a small training set with 500 samples has difficulties to capture all the information of a natural image, so we get poor results with this setting, particularly in the Dice metric. The results achieved with training sets of 1500 and 3000 samples are not



(a) Random sampling (b) Grid sampling

Figure 5.6: Segmentation results of OPF-Large-Data with 1500 training samples in one image of the GrabCut database. In (a) the training samples are chosen by random sampling, and in (b) the training samples are chosen by grid sampling.

so divergent, only a difference is appreciated according to the under-segmentation error in Figure 5.7b. The run-time experiment behaves as expected, an increase of the number of samples in the training set results in a clear increase of the execution time.

Is there a significant difference between the Exhaustive Search and the Local Search when looking for the adjacency parameter k in the OPF-based methods?

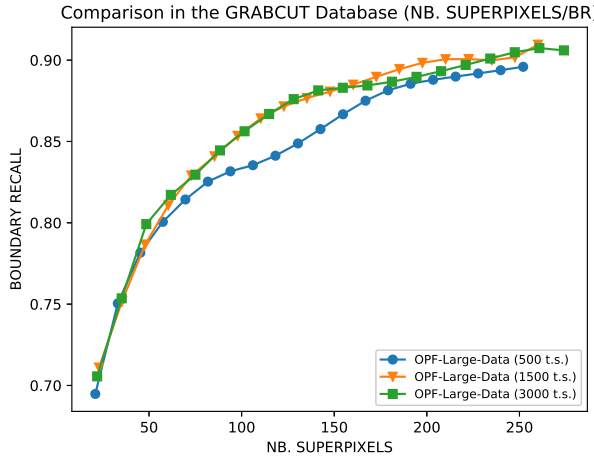
Figure 5.8 compares the segmentation results of OPF-Large-Data, in the GrabCut database, when using the exhaustive search and the local search to find the adjacency parameter k[7]. We execute both versions of OPF-Large-Data, with the same value of $k_{\max}$ for many values of $k_{\max}$.

It can be observed that the results are almost identical, both in boundary recall and under-segmentation error. This means that in most cases, both searches found the same value (or at least not so distant values) of k. Therefore, it seems a good idea to use this local search, instead of the exhaustive search, in the divide-and-conquer extensions (OPF-Blocks-1 and OPF-Blocks-2) to improve their performance, and so we do.

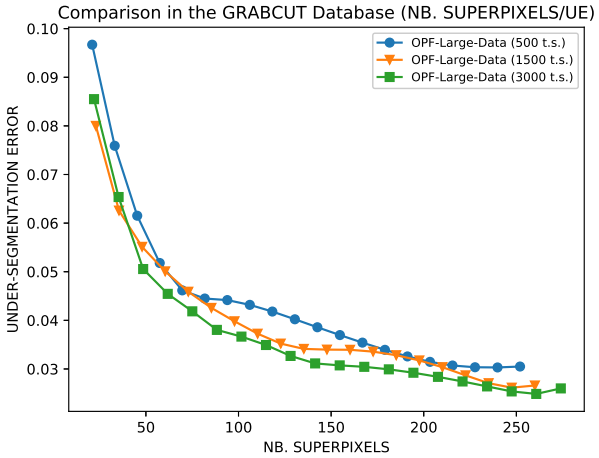
How many blocks are needed in the first level of the divide-and-conquer extensions?

Figures 5.9 and 5.10 compare the segmentation results of OPF-Blocks-1 and OPF-Blocks-2 in the GrabCut database considering different numbers of blocks for the first level

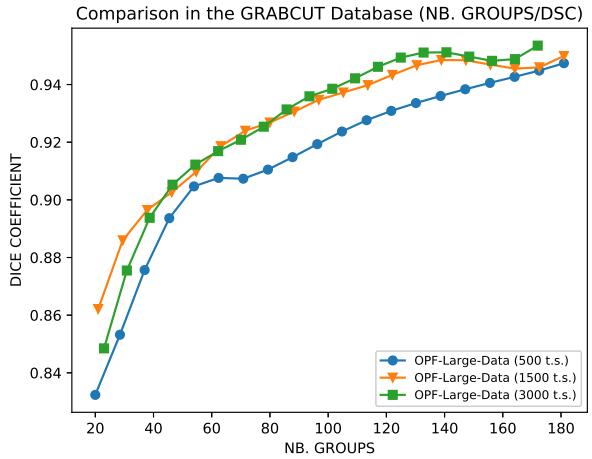
[7]Remember that the local search remains with the local minimum when evaluating the normalized cut function $C(k)$ (see Equation 3.8) within the interval $[1, k_{\max}]$ starting with $k_{\max}$. Instead, the exhaustive search test all values of $k \in [1, k_{\max}]$ and remains with the value that minimizes $C(k)$. For more details, see Section 4.2



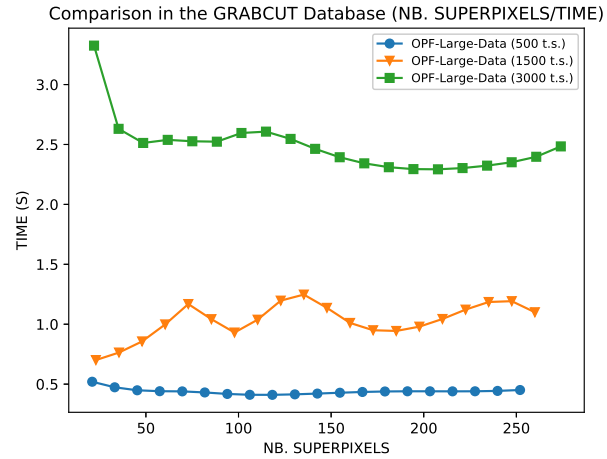
(a) Comparison according to the boundary recall metric.



(b) Comparison according to the under-segmentation error.



(c) Comparison according to the Dice coefficient.



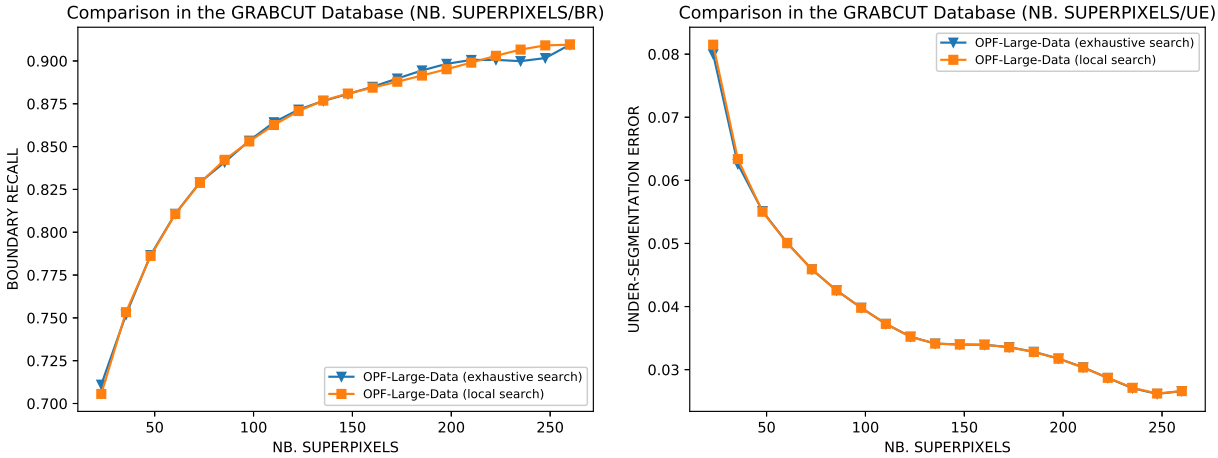
(d) Comparison according to the run-time performance.

Figure 5.7: Comparisons between the segmentation results obtained by OPF-Large-Data, in the GrabCut database, when training sets of different sizes are used.

(base level), respectively. In these experiments, each block is clustered with 1500 training samples chosen by grid sampling, therefore, more blocks imply a greater amount of training samples.

According to Figure 5.9, it does not seem a good idea to run OPF-Blocks-1 with many blocks unless our objective is to get a lot of superpixels. The more blocks we have, more superpixels we usually get. As clusters produced in different blocks cannot be merged directly because of the absence of a second level of clustering, the final result has a great dependence on the post-processing that tries to merge adjacent superpixels by comparing their histograms (see Section 4.3). Figure 5.9b shows that the under-segmentation error up to 130 superpixels is greater for 16 blocks than for both 4 and 9 blocks.

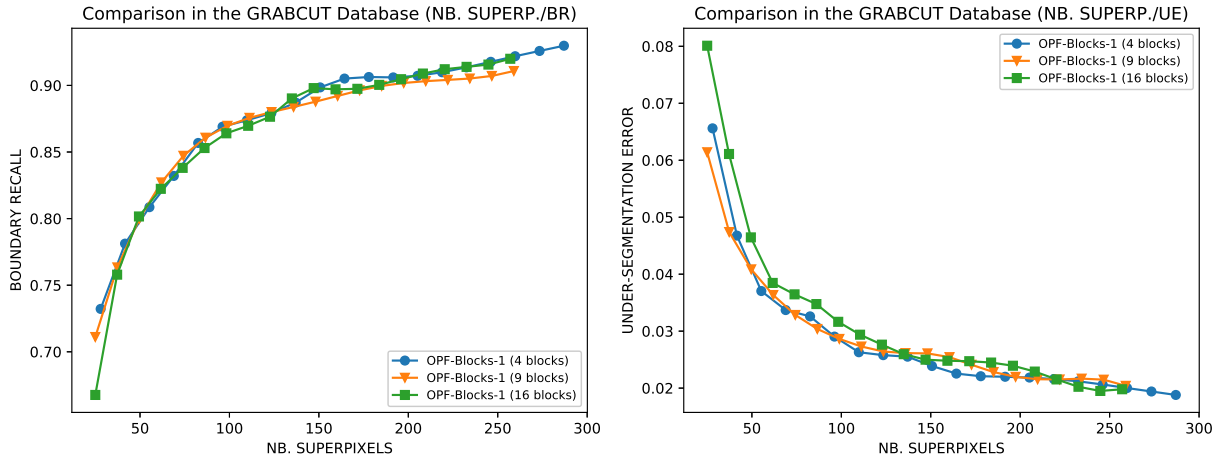
Unlike OPF-Blocks-1, OPF-Blocks-2 allows merging clusters formed in different blocks because it has two levels. According to Figure 5.10, the more blocks in the first level, the better the chance of improving the segmentation result. However, a large number



(a) Comparison according to the boundary recall metric.

(b) Comparison according to the under-segmentation error.

Figure 5.8: Comparisons between the segmentation results of OPF-Large-Data, in the Grabcut database, when different strategies to find the adjacency parameter k are used.



(a) Comparison according to the boundary recall metric.

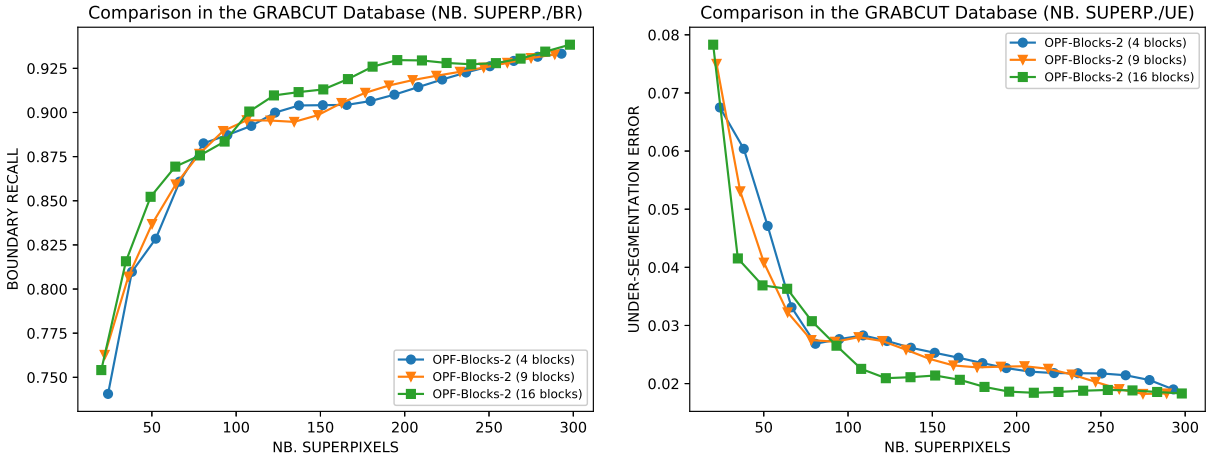
(b) Comparison according to the under-segmentation error.

Figure 5.9: Comparisons between the segmentation results of OPF-Blocks-1, in the GrabCut database, when a different number of blocks for the first level is established.

of blocks in the first level may be unnecessary depending on the size and entropy of the image. In addition, many blocks usually involve a degradation in the execution time of the technique.

Comparing the proposed technique with the baseline methods in the GrabCut database

Figures 5.11, 5.12, and 5.13 compare the segmentation results of the baseline methods and the two divide-and-conquer extensions in the images of the GrabCut database. It can be seen that OPF-Blocks-2 significantly outperforms the others techniques, according



(a) Comparison according to the boundary recall metric.

(b) Comparison according to the under-segmentation error.

Figure 5.10: Comparisons between the segmentation results of OPF-Blocks-2, in the GrabCut database, when a different number of blocks for the base level is established.

to the boundary recall and the under-segmentation error. OPF-Blocks-1 has the second best performance, only surpassed by OPF-Large-Data up to 50 superpixels. OPF-Large-Data overcomes SLIC, k-Means, Quick-Shift, and Watershed in boundary recall, but it is surpassed by these methods according to the under-segmentation error from 150 superpixels. Quick-shift has the worst performance up to 80 superpixels but from there, its result improves. Watershed is the least effective method, especially from 100 superpixels according to the boundary recall. Considering the run-time efficiency, Watershed and SLIC are superior to the others. The two methods never reach a second, regardless of the number of superpixels generated. OPF-Large-Data and the divide-and-conquer extensions have a similar performance, taking approximately 1.5 seconds to cluster each image. The performance of k-Means deteriorates as the amount of superpixels increases, reaching almost 10 seconds to produce about 300 superpixels per image. Instead, the performance of Quick-Shift improves as the number of superpixels augments (this is due to the decrease of the search space to estimate the PDF of the samples). These results (results corresponding to the execution time of the techniques) are equivalent in the three databases evaluated.

Figures 5.14, 5.15, 5.16, 5.17, 5.18, 5.19, and 5.20 exhibit segmentation results (with approximately 100 superpixels) of the compared methods in some images of the GrabCut database. The yellow arrows indicate leaks between the object and background. It can be seen that the superpixels generated by k-Means and SLIC are more regular and compact than those produced by the other methods. Quick-Shift, Watershed, and the OPF-based extensions create rather irregular superpixels in both size and shape. k-Means and Watershed are the procedures with worst boundary adherence in the images shown. SLIC, OPF-Large-Data, and Quick-Shift recover most boundaries of the objects; however, the divide-and-conquer extensions identify almost all. It can be seen that some superpixels produced by OPF-Blocks-1 and OPF-Blocks-2 have straight edges. This is a consequence

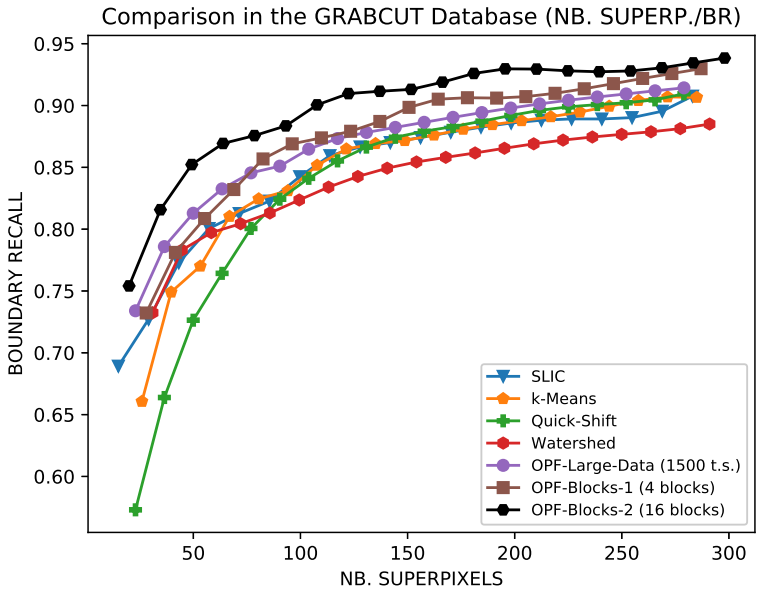


Figure 5.11: Comparison between the segmentation results of the methods, according to the boundary recall metric, in the GrabCut database.

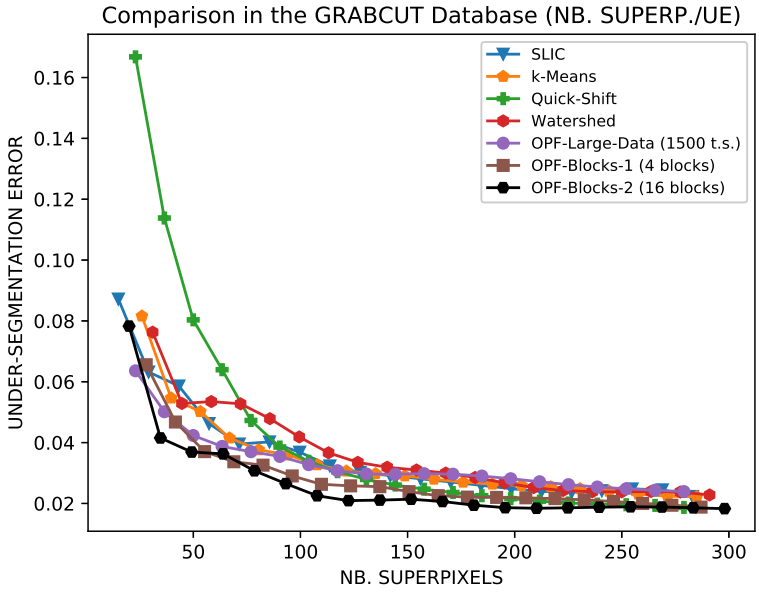


Figure 5.12: Comparison between the segmentation results of the methods, according to the under-segmentation error, in the GrabCut database.

of two adjacent clusters, discovered in different blocks of the first level, that do not come together afterward.

Figures 5.21 and 5.22 reveal comparisons between the clustering methods, according to the accuracy and Dice metrics, after true label propagation by each cluster representative. It can be observed that OPF-Large-Data has the best results up to 50 groups, while the divide-and-conquer extensions exceed all others techniques in both metrics from 60 groups. SLIC has the lowest performance in the Dice metric; however, it reaches the divide-and-

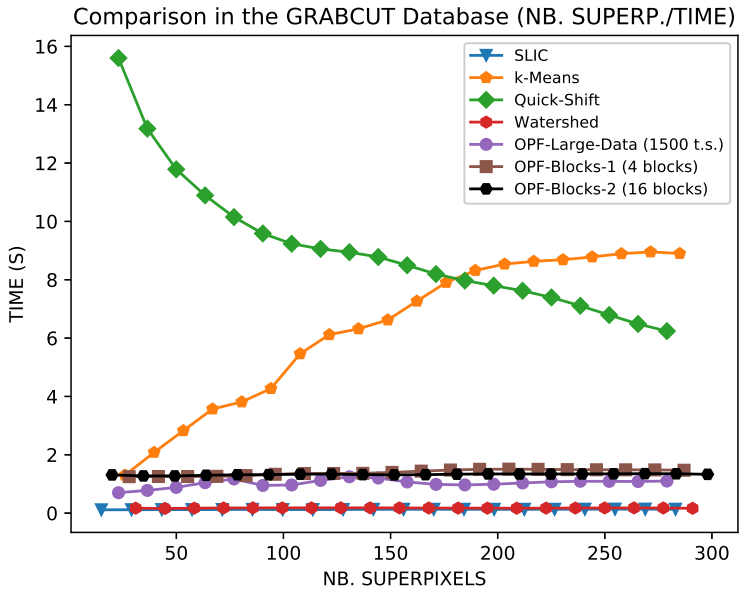


Figure 5.13: Comparison between the methods according to the execution time in the GrabCut database.

conquer extensions in accuracy from 150 groups.

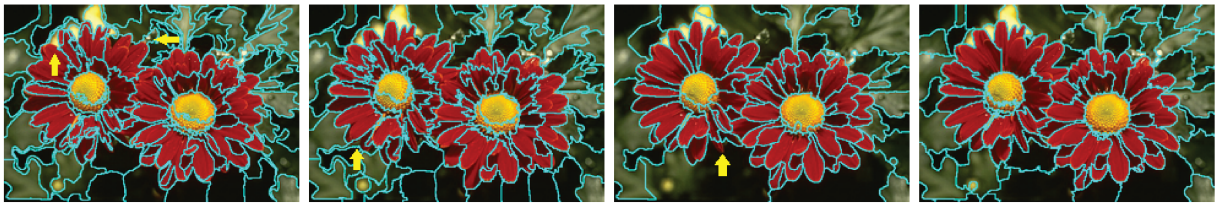
Figures 5.23, 5.24, 5.25, 5.26, 5.27, 5.28, 5.29, and 5.30 present some label propagation results, for about 100 clusters, after reducing noise by morphological operations and remaining with the largest object component. It can be seen that the divide-and-conquer extensions, especially OPF-Blocks-2, recover an object quite similar to the ground-truth object in all shown images. The other techniques have some problems to identify the object in at least two images. k-Means has problems in Figures 5.23, 5.26, 5.27, 5.28, and 5.29; SLIC gets trouble in Figures 5.24, 5.29, and 5.30; and OPF-Large-Data presents difficulties in Figures 5.26 and 5.30.

Comparing the proposed technique with the baseline methods in the Parasites/Impurities database

Figures 5.31 and 5.32 reveal comparisons between the segmentation results of the tested methods in the Parasites/Impurities database. It can be observed that OPF-Blocks-2 has the best performance both in boundary recall and the under-segmentation error. k-Means and OPF-Blocks-1 also obtain very good results, being the first capable of overcoming OPF-Blocks-2 in boundary recall from 180 superpixels. OPF-Large-Data and SLIC have similar behavior in both measures. Watershed is the technique with the worst performance and only beats Quick-Shift up to 60 superpixels.

Figures 5.33, 5.34, 5.35, 5.36, 5.37, 5.38, and 5.39 present some segmentation results of the compared methods with approximately 50 superpixels. In these images, the good boundary adherence of OPF-Blocks-2 to the parasites' boundaries can be verified. The other techniques fail in some cases to separate the parasites from the impurities.

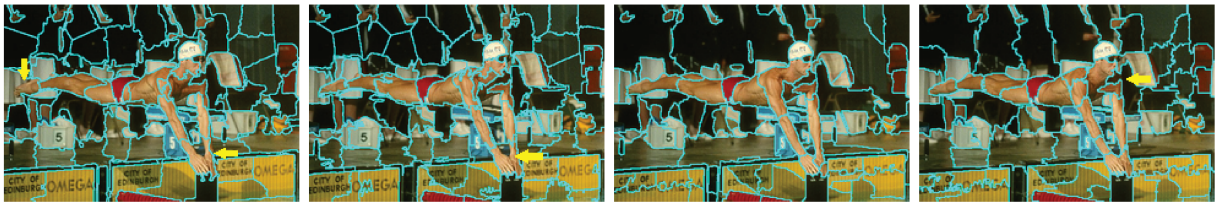
Figures 5.40 and 5.41 show comparisons between the clustering methods according to



(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift

(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

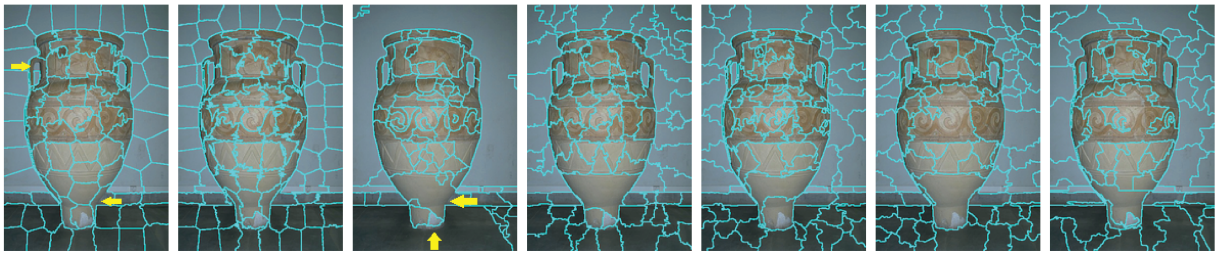
Figure 5.14: Segmentation results of the compared methods in the 2nd image of the GrabCut database.



(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift

(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.15: Segmentation results of the compared methods in the 4th image of the GrabCut database.



(a) k-Means (b) SLIC (c) Water-shed (d) Quick-Shift (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

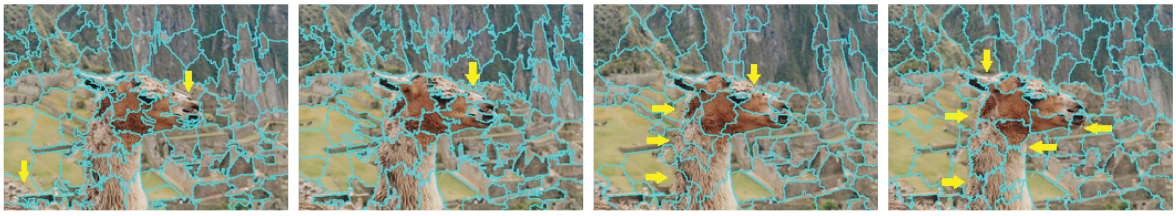
Figure 5.16: Segmentation results of the compared methods in the 10th image of the GrabCut database.

the accuracy and Dice metric after true label propagation from the cluster prototypes. It can be seen that OPF-Large-Data obtains the worst results in both metrics. OPF-Blocks-1 and SLIC have a similar performance from 120 groups, but up to this number of

(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift

(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

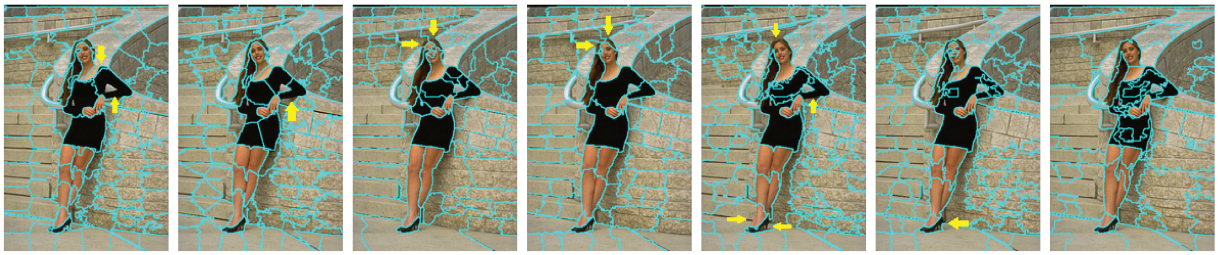
Figure 5.17: Segmentation results of the compared methods in the 25th image of the GrabCut database.



(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift

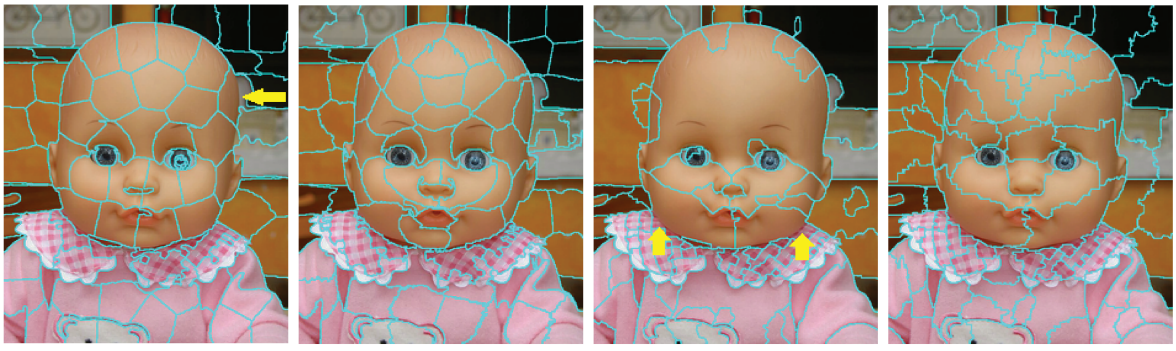
(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.18: Segmentation results of the compared methods in the 34th image of the GrabCut database.

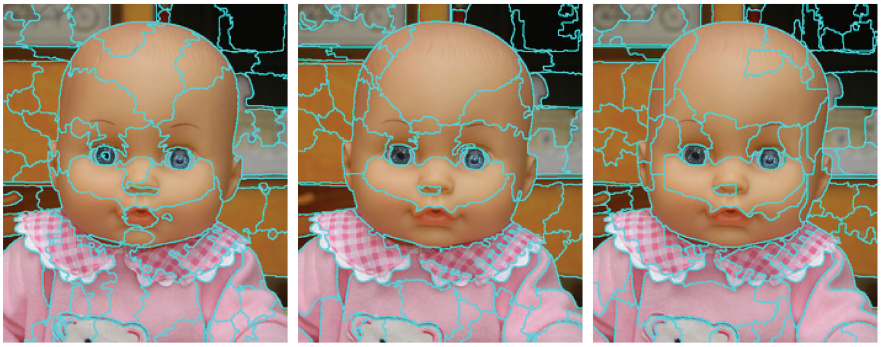


(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.19: Segmentation results of the compared methods in the 17th image of the GrabCut database.



(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift



(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.20: Segmentation results of the compared methods in the 29th image of the GrabCut database.

groups, the first surpasses the second in accuracy and the opposite happens in the Dice metric. OPF-Blocks-2 and k-Means get the best results, however, the second exceeds the first one up to 60 groups.

Figures 5.42, 5.43, 5.44, 5.45, 5.46, 5.47, and 5.48 exhibit some results of the methods after true label propagation from the cluster representatives, for approximately 20 groups. For such small number of clusters, it is normal that the compared techniques cannot recover the entire parasite in some cases. Therefore, k-Means has problems in Figures 5.43, 5.45, and 5.46; SLIC gets trouble in Figures 5.45 and 5.47; OPF-Large-Data has difficulties in Figures 5.42 and 5.45; OPF-Blocks-1 has problems in Figure 5.44; and OPF-Blocks-2 presents complications in Figures 5.45 and 5.46.

Comparing the proposed technique with the baseline methods in the Liver database

The graphics from Figures 5.49 and 5.50 compare the effectiveness of the methods, according to superpixel quality metrics, when segmenting the images of the Liver database. In these experiments, we discard all the dark pixels because we know they are part of the background. It can be observed that Quick-Shift has the worst performance by far in both metrics. OPF-Blocks-1, OPF-Blocks-2, and Watershed exceed all other techniques in under-segmentation error, while OPF-Large-Data, SLIC, and k-Means get similar results according to this metric. The results corresponding to the boundary recall metric are different. Watershed has the second-worst performance while the divide-and-conquer

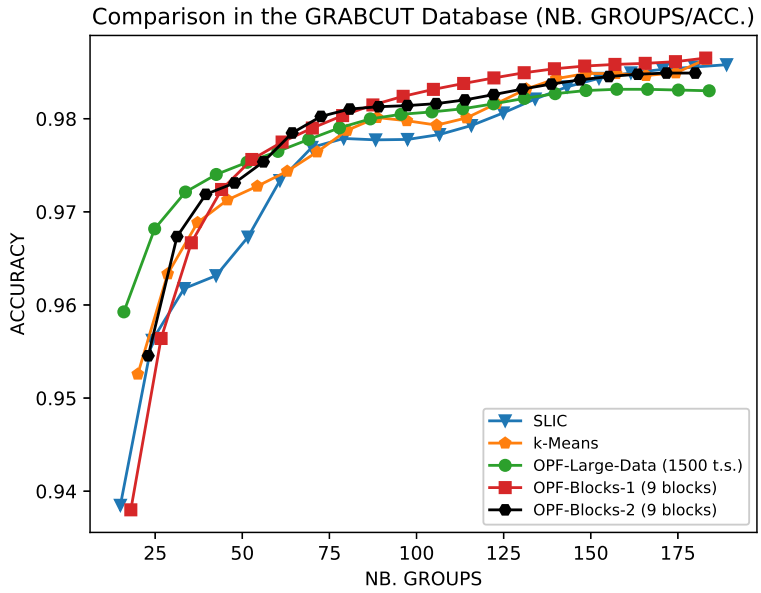


Figure 5.21: Comparison between the segmentation results of the methods after true label propagation from the cluster prototypes, according to the accuracy, in the GrabCut database.

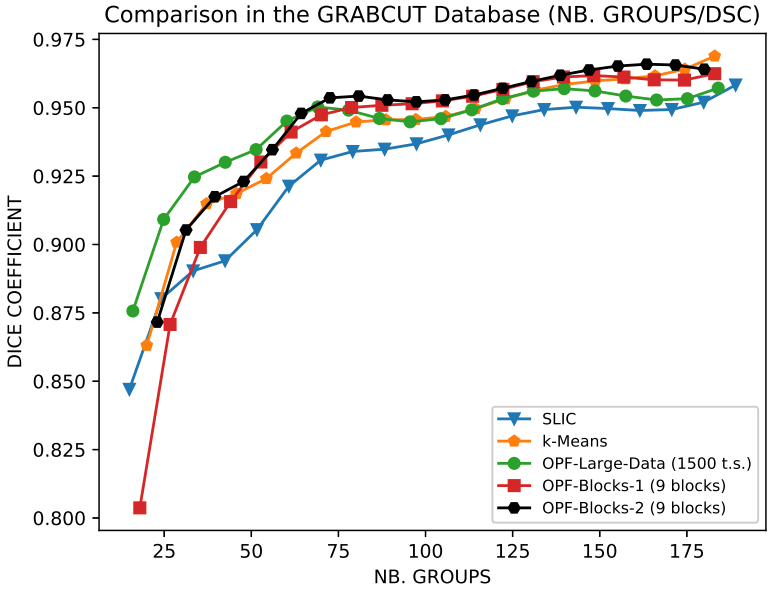
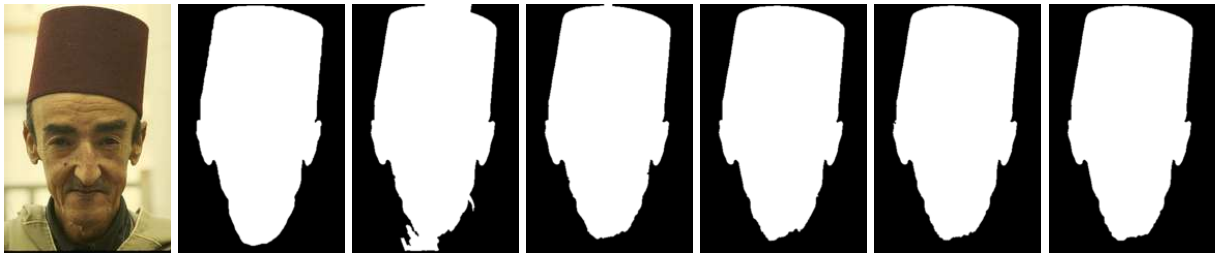


Figure 5.22: Comparison between the segmentation results of the methods after true label propagation from the cluster prototypes, according to the Dice coefficient, in the GrabCut database.

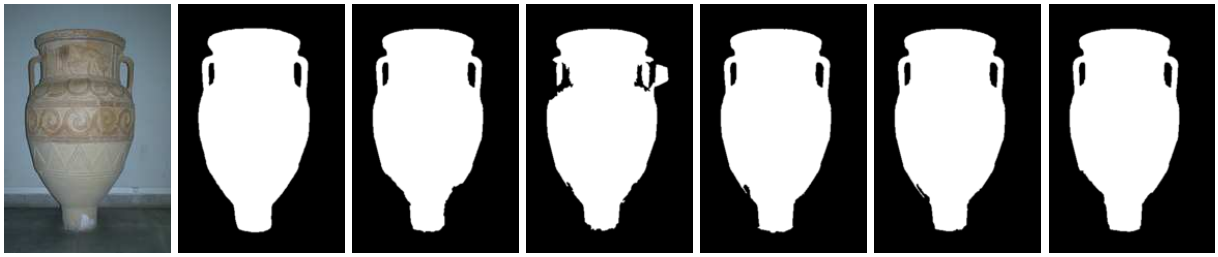
extensions surpass SLIC and OPF, but are exceeded by k-Means from 100 superpixels.

Figures 5.51, 5.52, 5.53, 5.54, 5.55, 5.56, and 5.57 show some segmentation results (with approximately 30 superpixels) of the compared techniques. It can be seen that the superpixels generated by Quick-Shift do not respect the boundaries of the liver and the other organs. k-Means, SLIC, and OPF-Large-Data generate superpixels quite irregular in both size and shape. In addition, these methods cannot segment the liver in most of the images. Instead, Watershed, OPF-Blocks-1, and OPF-Blocks-2 generate regular super-



(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.23: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 6th image of the GrabCut database.



(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.24: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 10th image of the GrabCut database.

(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.25: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 15th image of the GrabCut database.

(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.26: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 17th image of the GrabCut database.

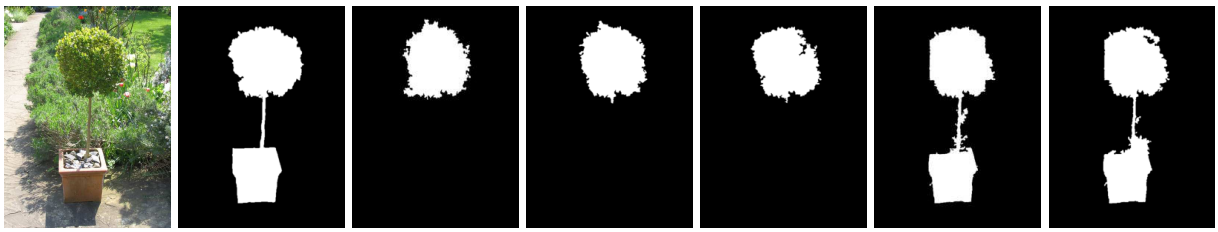
pixels according to the body components of the computed tomography slices. However, Watershed also has problems to delineate the liver boundaries. The superpixels generated by OPF-Blocks-1 and OPF-Blocks-2 allow separating the liver from the other organs in

(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.27: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 19th image of the GrabCut database.

(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.28: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 21st image of the GrabCut database.



(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.29: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 26th image of the GrabCut database.

(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.30: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 45th image of the GrabCut database.

an almost perfect way.

Figures 5.58 and 5.59 present comparisons of the segmentation results of the methods, after true label propagation from the cluster prototypes, in the images of the same database. It can be observed that OPF-Blocks-2 has the best performance in both the Dice coefficient and the accuracy, especially in the first metric. OPF-Blocks-1 has the second best effectiveness from 50 groups. The results of SLIC are not among the best; however, this technique is able to overcome the divide-and-conquer extensions in the accuracy metric from 120 superpixels. OPF-Large-Data gets the worst outputs from 75 groups.

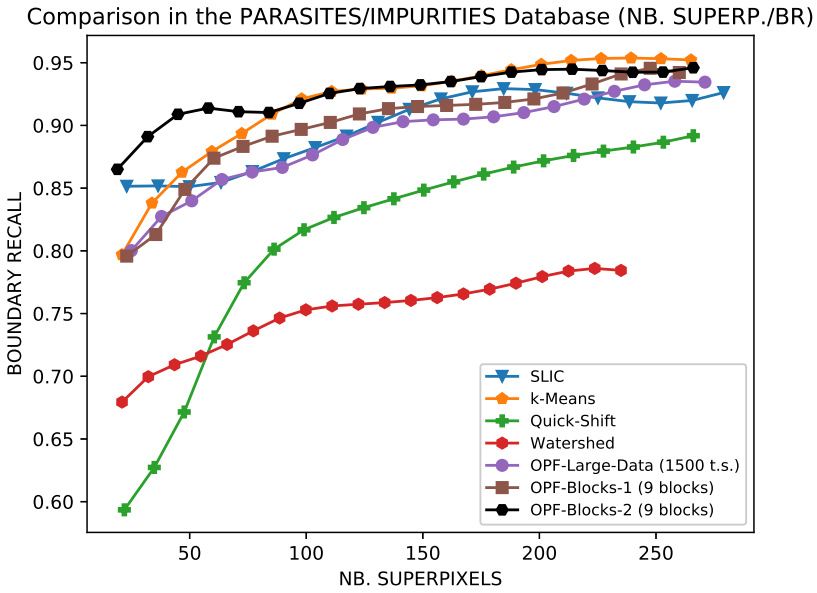


Figure 5.31: Comparison between the segmentation results of the methods, according to the boundary recall metric, in the Parasites/Impurities database.

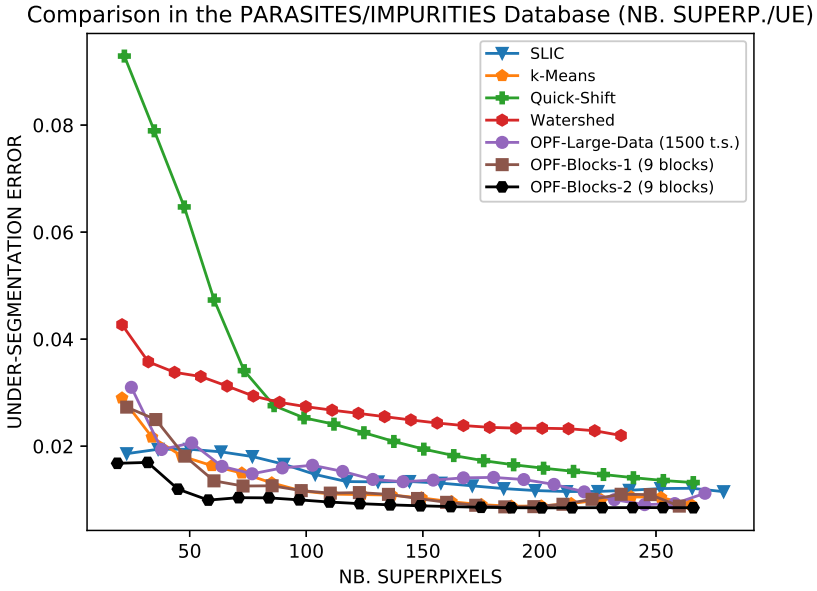
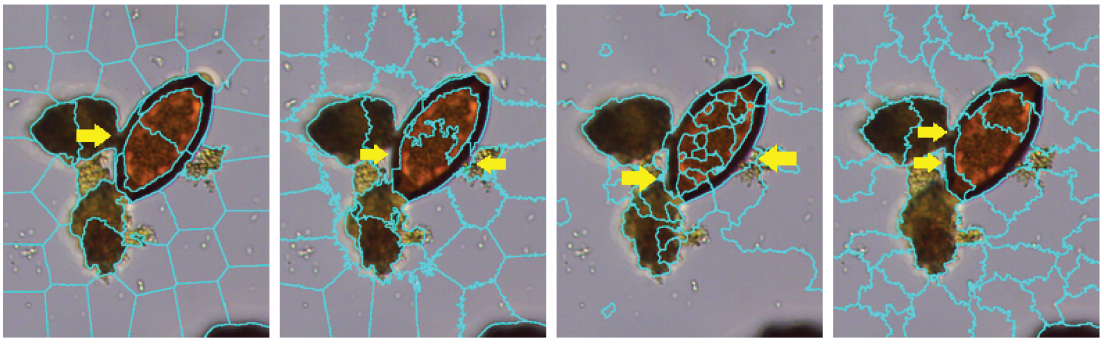
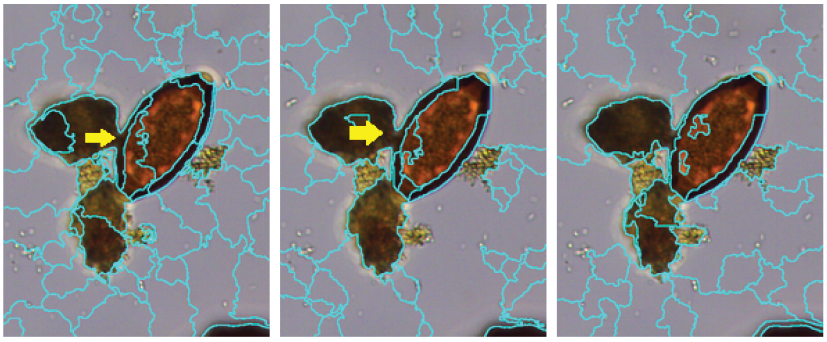


Figure 5.32: Comparison between the segmentation results of the methods, according to the under-segmentation error, in the Parasites/Impurities database.

Some label propagation results, from 50 to 60 groups, are shown in Figures 5.60, 5.61, 5.62, 5.63, 5.64, 5.65, and 5.66. In these images, it can be observed that k-Means and SLIC have problems to segment the liver in most cases. The results of OPF-Large-Data are not so bad, but the method has problems recovering the liver in Figures 5.60, 5.62, and 5.64. Instead, OPF-Blocks-1 and OPF-Blocks-2 get a good segmentation of the liver in all the images.

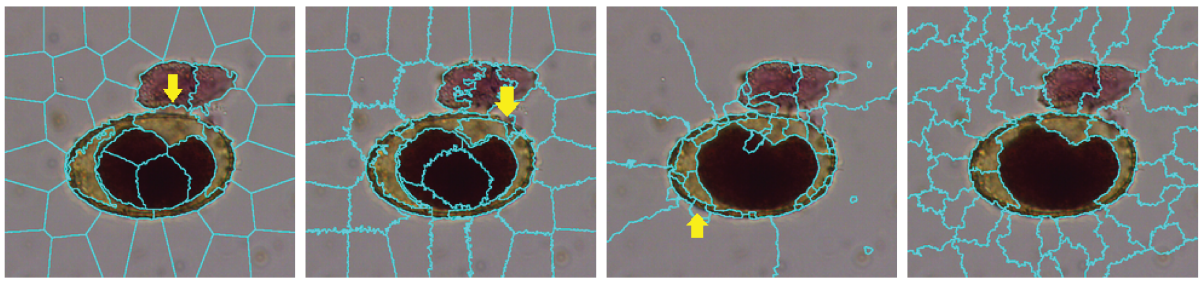


(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift

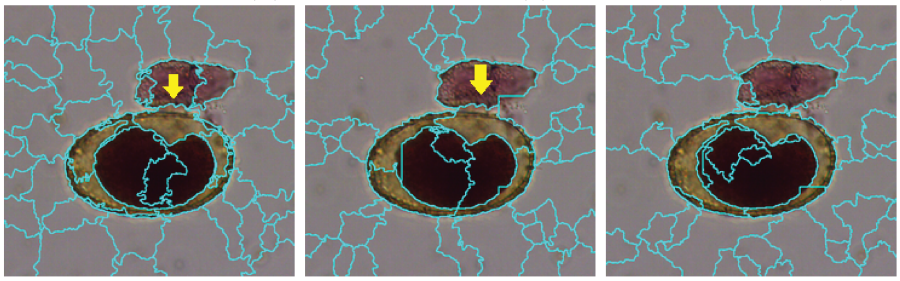


(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.33: Segmentation results of the compared methods in the 8th image of the Parasites/Impurities database.

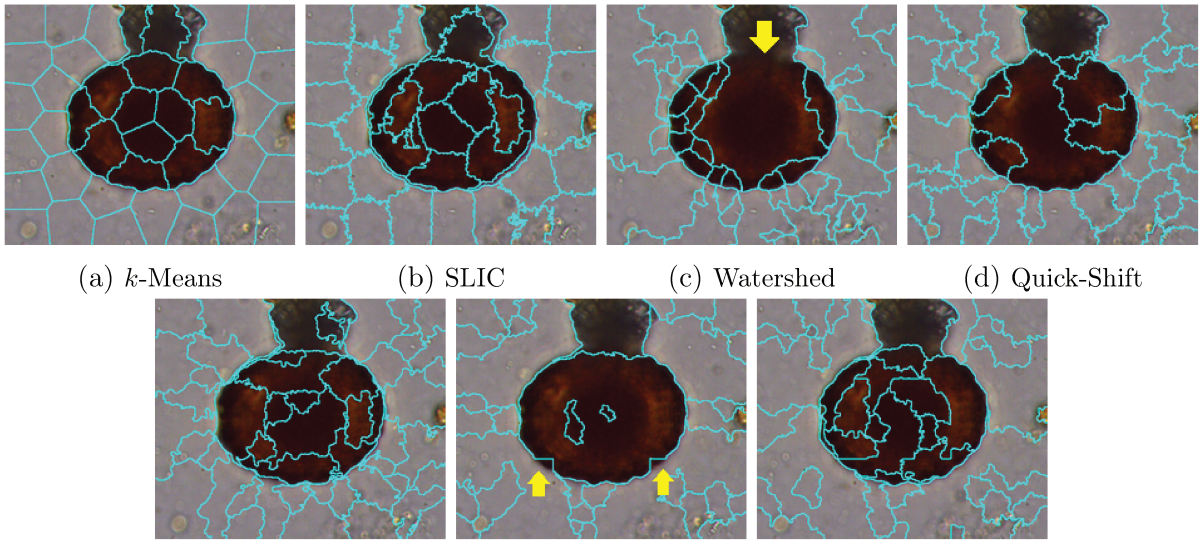


(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift



(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

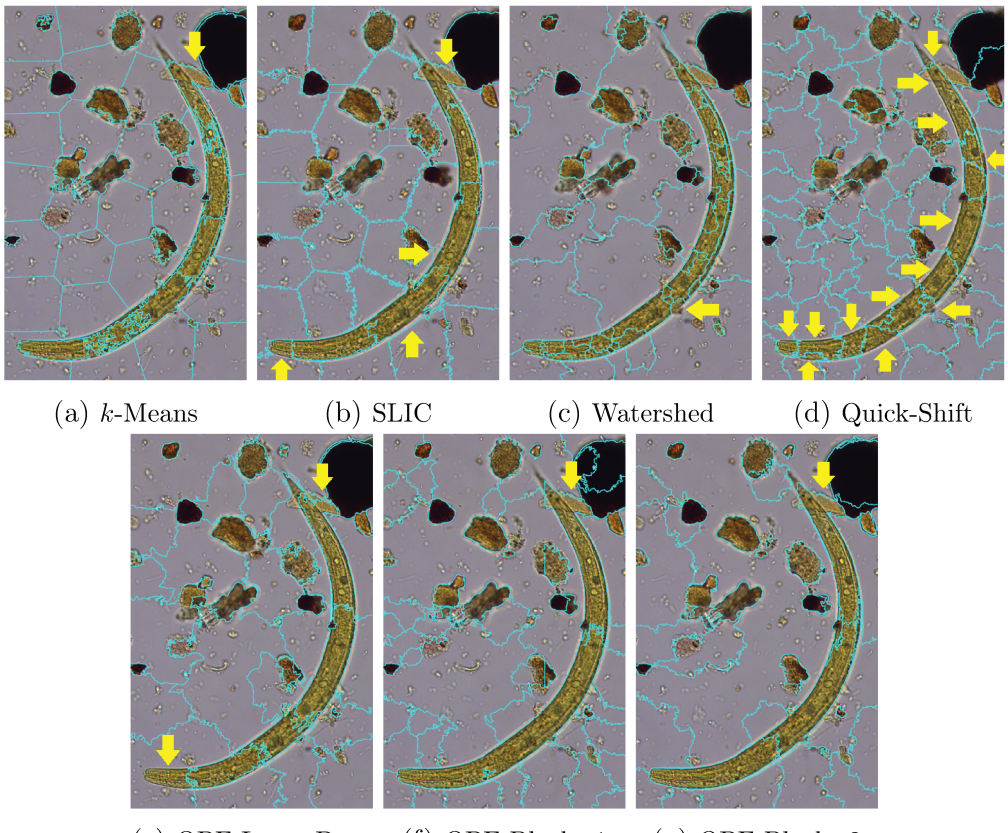
Figure 5.34: Segmentation results of the compared methods in the 15th image of the Parasites/Impurities database.



(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift

(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.35: Segmentation results of the compared methods in the 17th image of the Parasites/Impurities database.



(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift

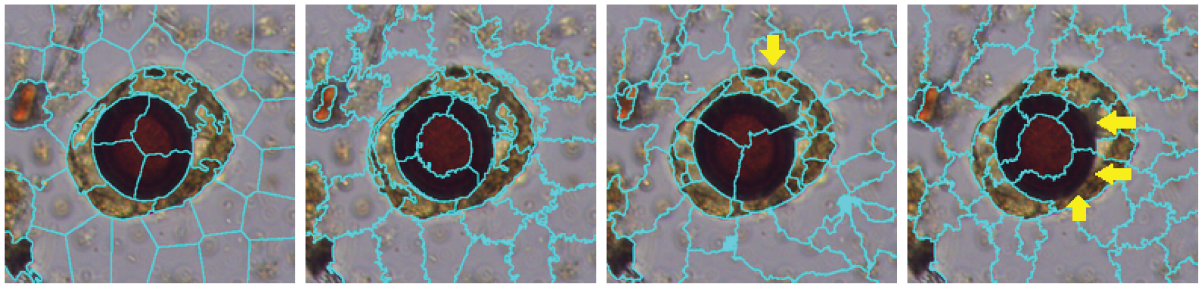
(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.36: Segmentation results of the compared methods in the 18th image of the Parasites/Impurities database.

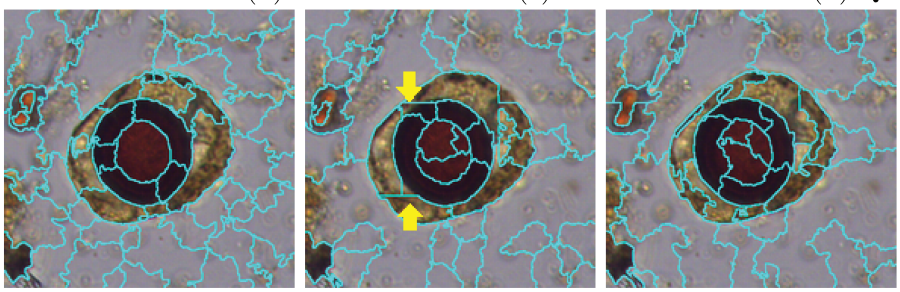
5.2 Clustering arbitrary datasets

In this section, we compare OPF-Blocks-2 against the original OPF clustering technique[8], OPF-Large-Data, and k-Means in six real-world datasets. Four of these datasets are

[8]We mean to the technique where Algorithm 1 is executed in the graph formed by all samples of the dataset. In this case, the training set is the entire dataset.

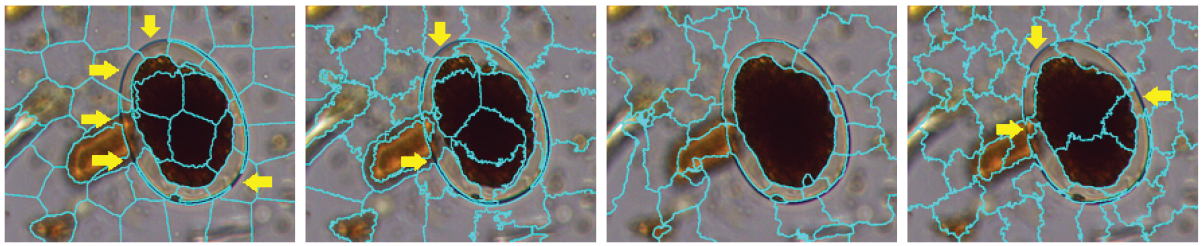


(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift

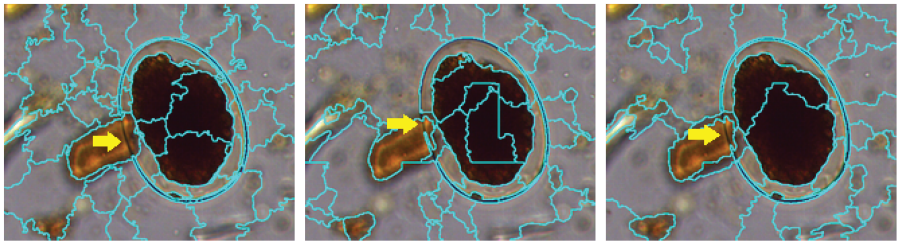


(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.37: Segmentation results of the compared methods in the 19th image of the Parasites/Impurities database.



(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift



(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

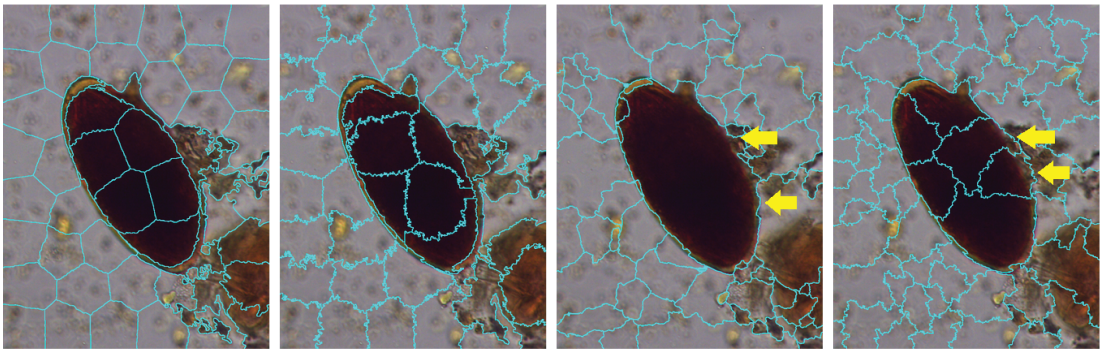
Figure 5.38: Segmentation results of the compared methods in the 22nd image of the Parasites/Impurities database.

publicly available for research and the other two are private.

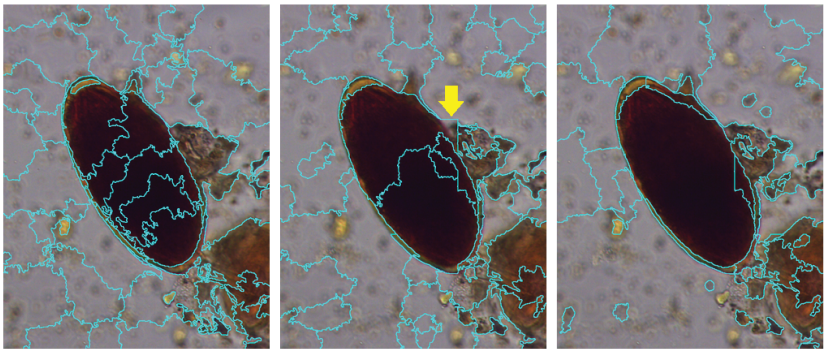
5.2.1 Datasets

Here we give a brief description of each tested dataset.

- **Pen-Based Recognition of Handwritten Digits Dataset (PenDigits):** A digit dataset that collects 250 examples from 44 writers. This dataset has 10,992 instances, 16 attributes per instance, and 10 classes representing the 10 digits in the



(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift



(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.39: Segmentation results of the compared methods in the 24th image of the Parasites/Impurities database.

decimal numeral system. The number of instances by class is rather similar. All classes have more than 1000 samples. For a more detailed description, it is available at `https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits`.

- **Protozoans Dataset (Protozoans):** A private dataset collecting 14,000 instances from the six most common protozoan species (intestinal parasites) in Brazil. The descriptor of the samples has 260 attributes, and it is based on the color, shape, and texture of the parasites[9]. The classes are not evenly distributed in the data. The species have 1576, 1320, 4855, 3558, 2858, and 89 samples, respectively. Figure 5.67 displays some examples of the protozoan species.

- **Eggs Dataset (Eggs):** A private dataset containing 3,578 instances from the eight most common species of human helminth eggs (intestinal parasites) in Brazil. The descriptor is the same as that of the Protozoans dataset. The classes are not uniformly distributed in the data. The species have 470, 144, 411, 114, 1191, 538, 242, and 468 samples, respectively. Some examples of the egg species can be seen in Figure 5.68.

- **Rome Superpixels Dataset (RomeSuperpixels):** This dataset is formed by 24,968 superpixels from a high resolution (VHR) image acquired in the Vatican City, in April 2004. A portion of the VHR image can be observed in Figure 5.69.

[9]For more information about this descriptor, it is described in [116].

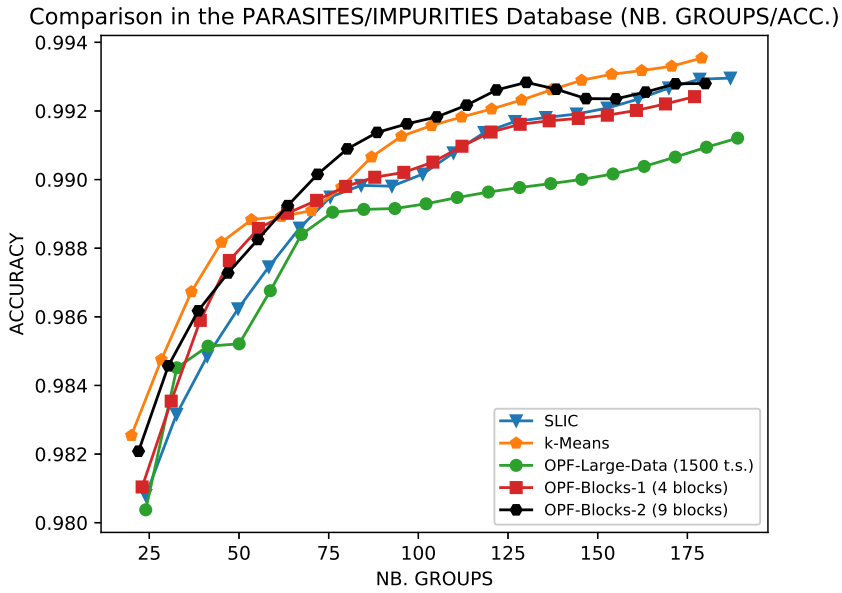


Figure 5.40: Comparison between the segmentation results of the methods after true label propagation from the cluster prototypes, according to the accuracy, in the Parasites/Impurities database.

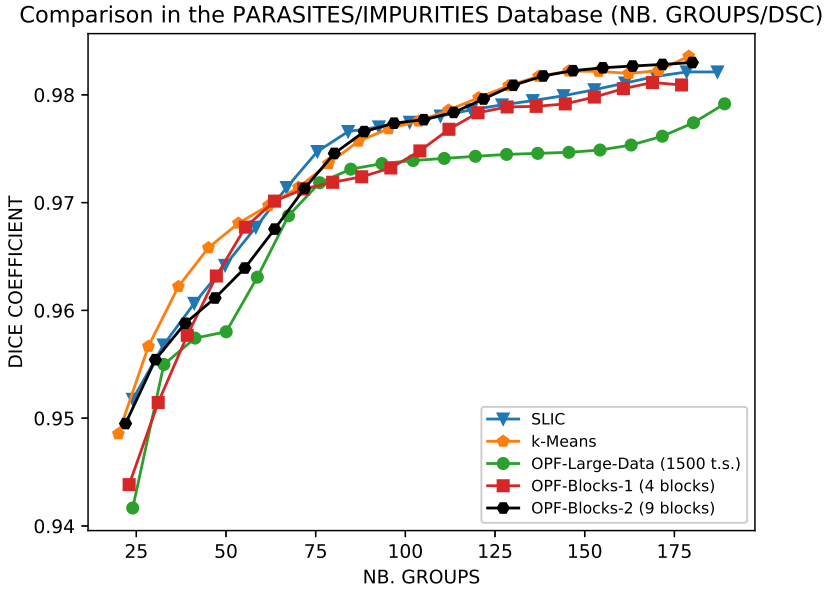
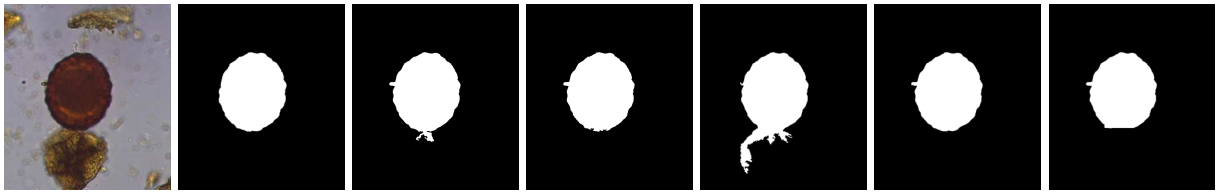


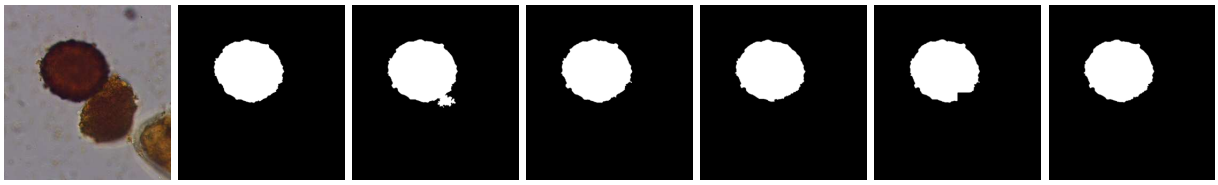
Figure 5.41: Comparison between the segmentation results of the methods after true label propagation from the cluster prototypes, according to the Dice coefficient, in the Parasites/Impurities database.

This image is labeled with seven classes of interest: road (2,048 samples), tree (2,936 samples), shadow (4,702 samples), water (843 samples), building (13,082 samples), grass (1,021 samples), and bare soil (336 samples). The feature vector used to describe the superpixels has 1795 attributes and is a combination of four descriptors: mean color, color histogram, local binary patterns (LBP), and border/interior clas-



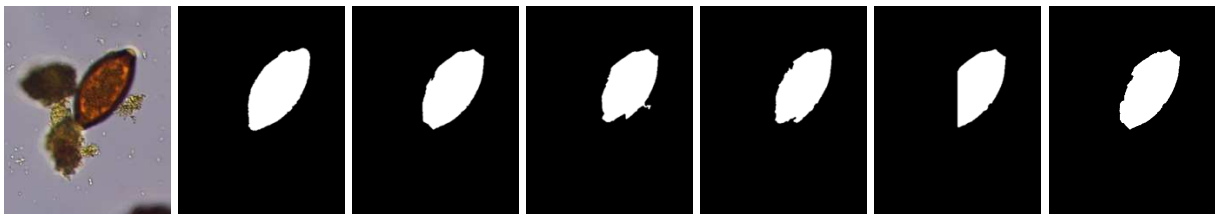
(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.42: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 1st image of the Parasites/Impurities database.



(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.43: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 3rd image of the Parasites/Impurities database.



(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.44: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 8th image of the Parasites/Impurities database.

(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

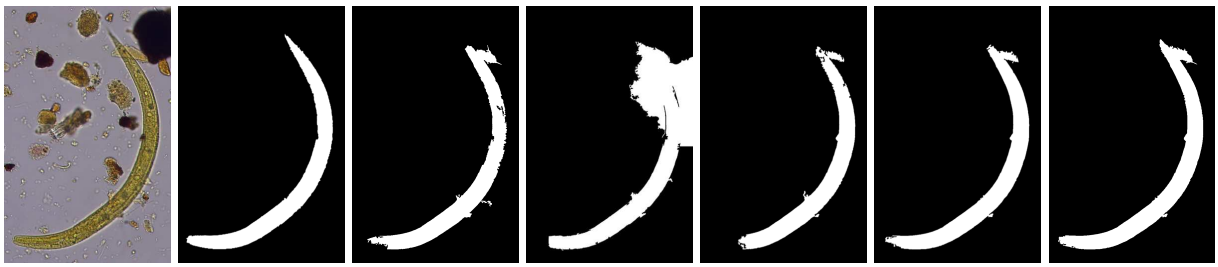
Figure 5.45: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 9th image of the Parasites/Impurities database.

sification (BIC). For a more detailed description, this dataset is described in [120].

- **Skin Segmentation Dataset (SkinSegmentation):** This dataset is built over the RGB color space from face images of various age groups (young, middle, and old), race groups (white, black, and Asian), and genders. It has 245,057 instances

(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.46: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 13th image of the Parasites/Impurities database.



(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.47: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 18th image of the Parasites/Impurities database.

(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.48: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 24th image of the Parasites/Impurities database.

where 50,859 are skin samples (skin pixels) and 194,198 are non-skin samples (non-skin pixels). For a more detailed description, it is available at `https://archive.ics.uci.edu/ml/datasets/skin+segmentation`.

- **Letter Recognition Dataset (LetterRecognition):** A dataset consisting of 20,000 black-and-white rectangular images, each corresponding to one of the 26 capital letters of the English alphabet. The descriptor has 16 attributes and the classes (capital letters) are fairly evenly distributed in the data. For a more detailed description, it is available at `https://archive.ics.uci.edu/ml/datasets/letter+recognition`.

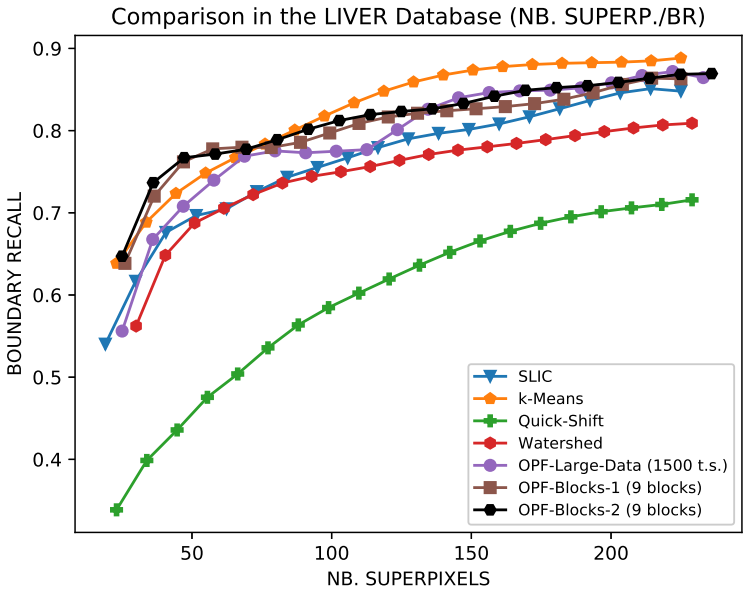


Figure 5.49: Comparison between the segmentation results of the methods, according to the boundary recall metric, in the Liver database.

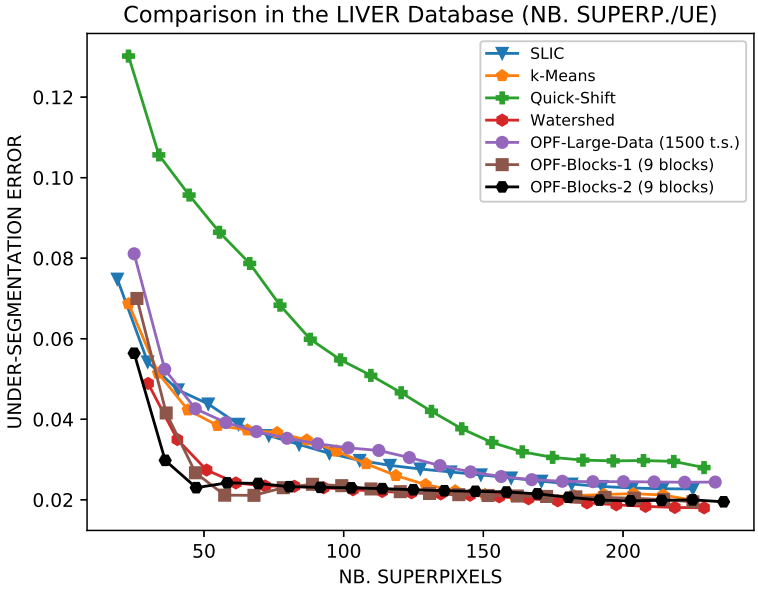
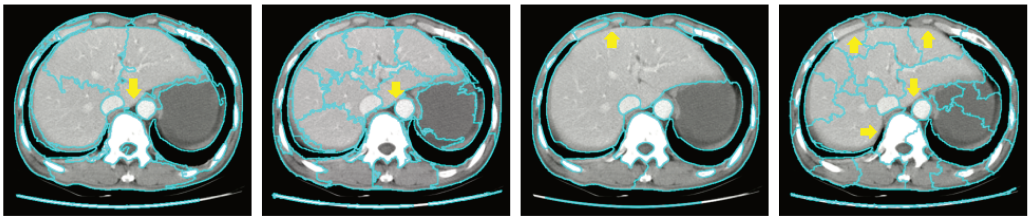


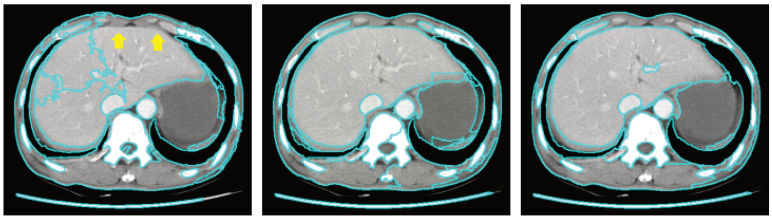
Figure 5.50: Comparison between the segmentation results of the methods, according to the under-segmentation error, in the Liver database.

5.2.2 Experimental methodology

OPF-based methods find natural groups in a dataset, but they do not guarantee a desired number of clusters. With these techniques, the number of clusters is dependent on the observation scale or $k_{\max}$ parameter. Other clustering methods, such as k-Means, can produce a specific number of groups. A problem is that, usually, the number of natural groups in a dataset is unknown.

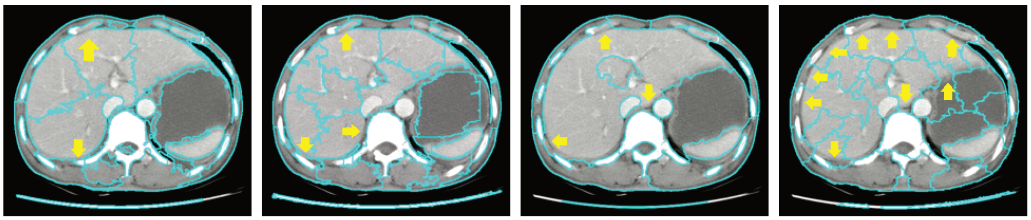


(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift

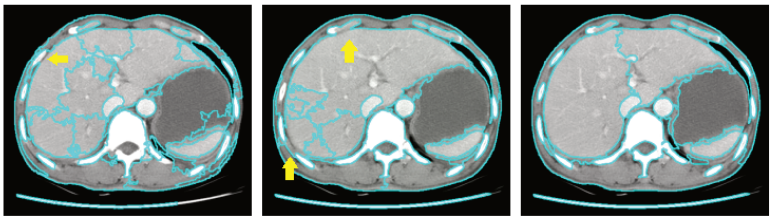


(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.51: Segmentation results of the tested methods in the 1st image of the Liver database.

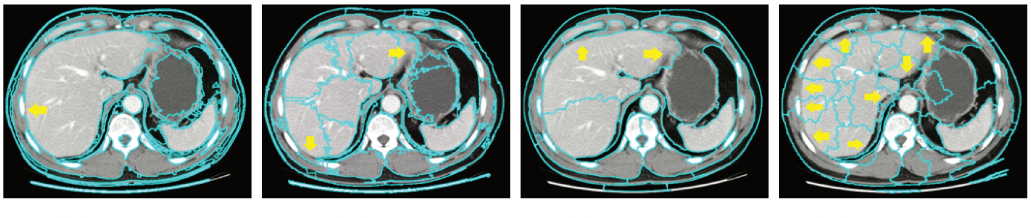


(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift

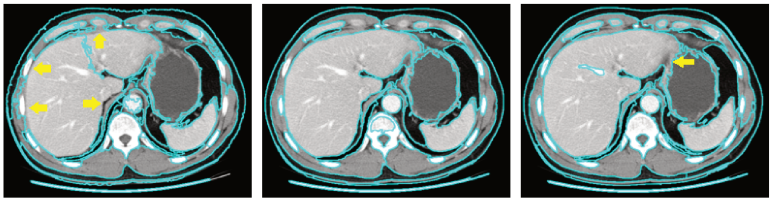


(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.52: Segmentation results of the tested methods in the 6th image of the Liver database.

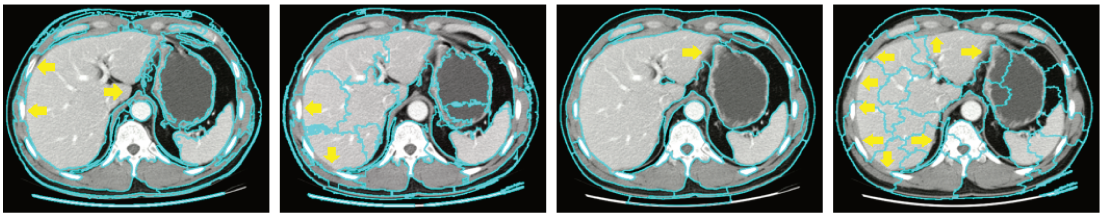


(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift

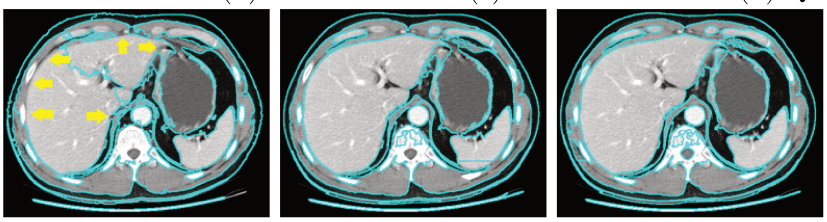


(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.53: Segmentation results of the tested methods in the 15th image of the Liver database.

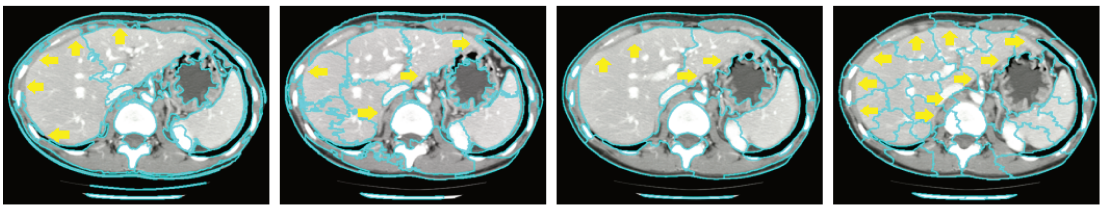


(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift

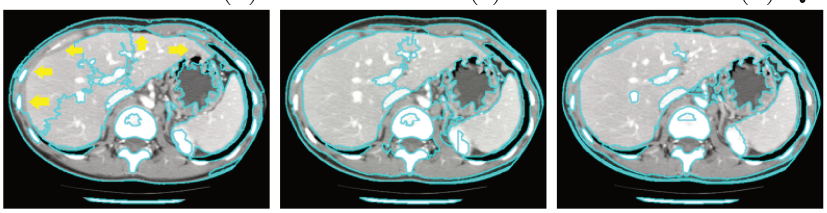


(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.54: Segmentation results of the tested methods in the 17th image of the Liver database.

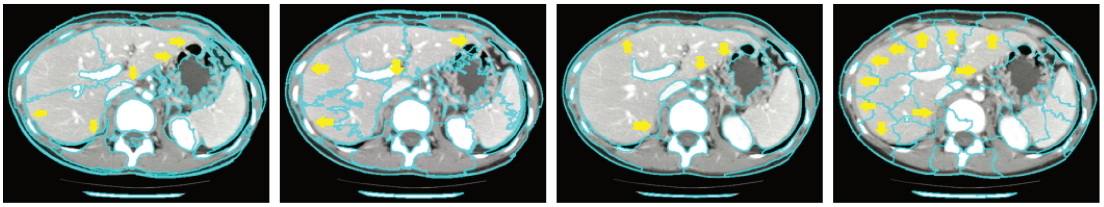


(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift

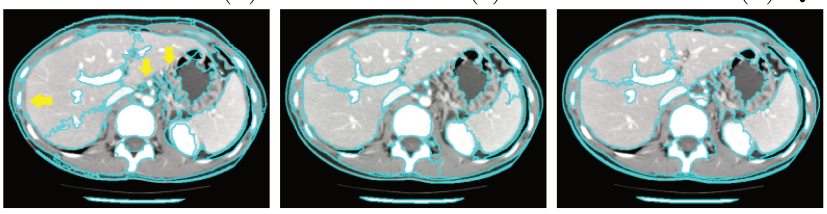


(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.55: Segmentation results of the tested methods in the 21st image of the Liver database.

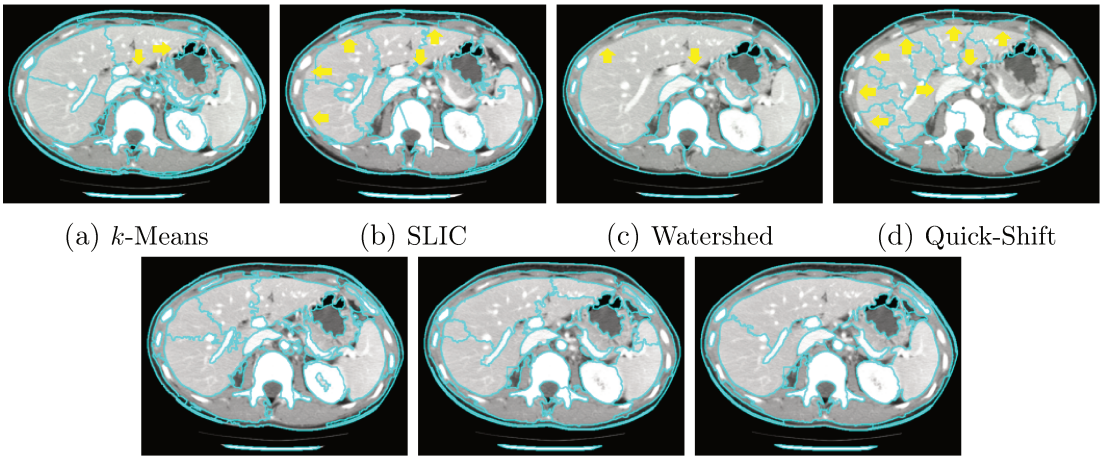


(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift



(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.56: Segmentation results of the tested methods in the 23rd image of the Liver database.



(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift

(e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.57: Segmentation results of the tested methods in the 28th image of the Liver database.

OPF-Blocks-2 allows many configurations when clustering data. It can divide all the samples into many or few blocks, or it can have a large or a small number of samples in the second level. It can also divide the dataset into big partitions and cluster each one by the OPF-Large-Data technique. However, the idea is to always use a configuration that does not compromise the efficiency (execution time) of the method. Following this suggestion in the experiments, we split up each dataset (except the SkinSegmentation dataset) in a way that all blocks, including the block formed by the samples of the second level, have about the same number of samples. This number cannot be very large, so it is chosen to be no greater than the number of training samples of OPF-Large-Data for each compared dataset — i.e., in a hypothetical experiment, we would compare OPF-Large-Data with 2000 training samples against OPF-Blocks-2 with no more than 2000 samples in each block. For the SkinSegmentation dataset, we divide the data into big partitions in the first level and cluster each one by OPF-Large-Data.

To compare the clustering techniques we reuse the idea of true label propagation from the cluster prototypes explained in Section 5.1.3. In this way, the accuracy is computed by fixed numbers of groups in each dataset. These predetermined numbers of clusters have a direct relationship to the number of classes in the datasets (e.g., twice the number of classes, three times, etc). In addition, we verify if the methods lose some classes after the true label propagation phase. A class is lost if none of the cluster prototypes has that class as true label. Ideally, an effective method must obtain high accuracy without losing classes. We also do the experiments backward. Given an established accuracy, we compute the number of clusters each method would need to reach it without losing any class. The smaller the number of groups, the better is the method. We also compare the methods according to their efficiency (time execution) in each dataset.

In each dataset, OPF-Large-Data, k-Means, and OPF-Blocks-2 are executed 50 times. The results of the experiments show the mean and the standard deviation of the accuracy for all iterations. OPF is only executed once in each dataset because its outcome is deterministic. Furthermore, for most datasets, there are shown the outcomes of OPF when the suggested local search to find the adjacency parameter k is used.

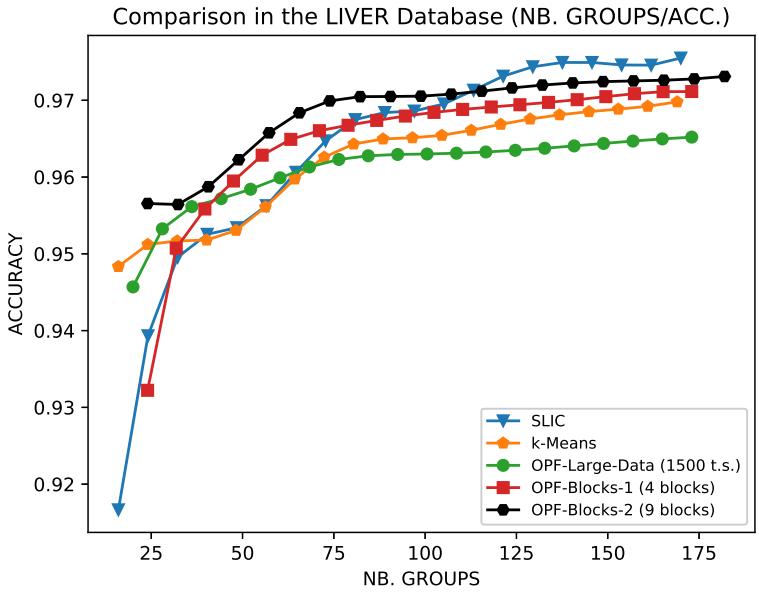


Figure 5.58: Comparison of the segmentation results of the methods after true label propagation from the cluster prototypes, according to the accuracy, in the Liver database.

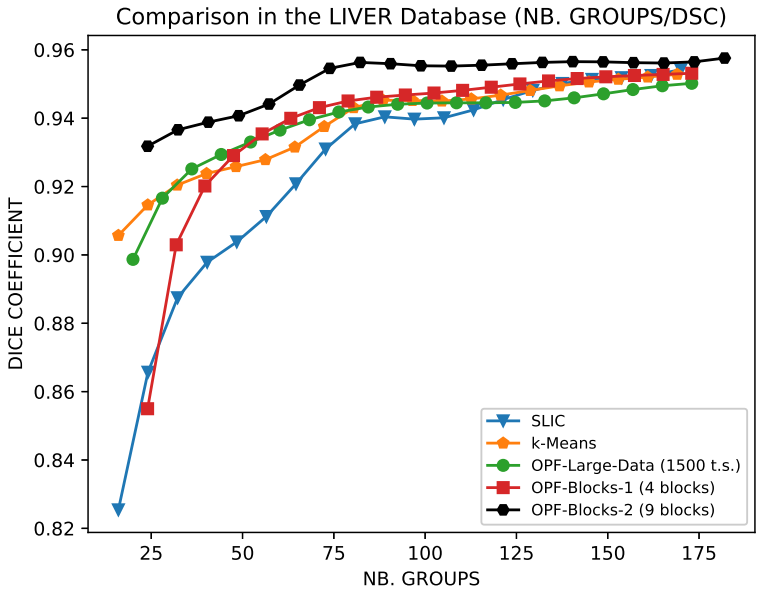


Figure 5.59: Comparison of the segmentation results of the methods after true label propagation from the clluster prototypes, according to the Dice coefficient, in the Liver database.

5.2.3 Results

Below, we present the results of the compared techniques in the tested datasets.

Comparisons in the PenDigits dataset

Table 5.1 shows the experimental results in the PenDigits dataset. As the dataset has 10 classes, we measure the accuracy of the methods for 20 and 30 groups — i.e., considering

(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.60: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 5th image of the Liver database.

(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.61: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 6th image of the Liver database.

(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.62: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 13th image of the Liver database.

(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.63: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 16th image of the Liver database.

(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.64: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 20th image of the Liver database.

2 and 3 groups per class. The OPF method achieves the best accuracy for both group numbers. OPF-Large-Data (with 2000 training samples) and OPF-Blocks-2 (with 6 blocks

(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.65: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 24th image of the Liver database.

(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Figure 5.66: Resulting mask of the compared methods, after true label propagation from the cluster prototypes, in the 26th image of the Liver database.

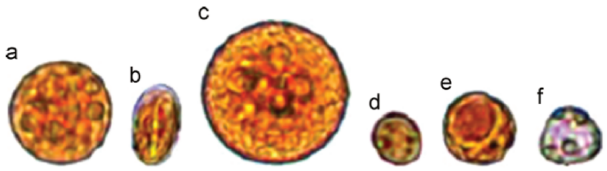


Figure 5.67: Some examples of the most common protozoan species. (a) Entamoeba histolytica/E. dispar, (b) Giardia duodenalis, (c) Entamoeba coli, (d) Endolimax nana, (e) Iodameba bütschlii, and (f) Blastocystis hominis.

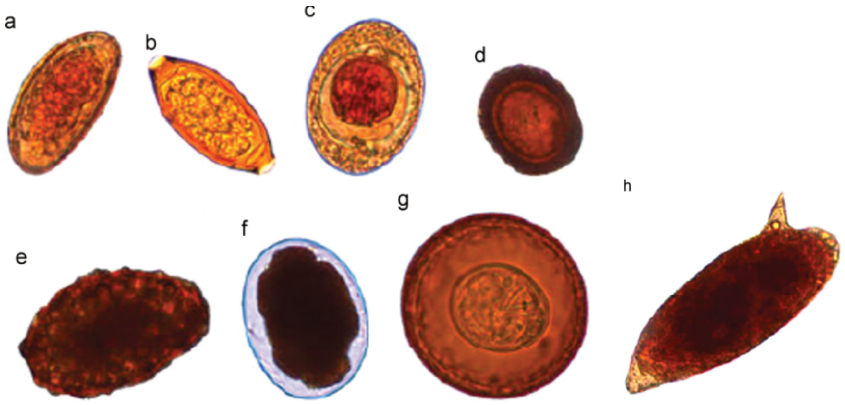


Figure 5.68: Some examples of the most common helminth species. (a) Enterobius vermicularis, (b) Trichuris trichiura, (c) Hymenolepis nana, (d) Taenia spp., (e) Ascaris lubmricoides, (f) Ancylostomatidae, (g) Hymenolepis diminuta, (h) Schistosoma mansoni.

in the first level) obtain similar results, although the latter gets a lower standard deviation. k-Means is the method with the lowest accuracy; however, it is the fastest. It can be seen that the use of the local heuristic to find the adjacency parameter k allows reducing the execution time of OPF from 4.7 to 1.3 seconds when generating 20 groups. We also compute the number of clusters that each method would need to reach an accuracy value of 0.90. Again, OPF is the best method just needing 22 groups to achieve the goal, while

Figure 5.69: A portion of the aerial image of Rome with the corresponding superpixels.

k-Means is the worst needing 36 groups.

Table 5.1: Experimental results in the PenDigits dataset.

Method	**Accuracy/Time (20 Groups)**	**Accuracy/Time (30 Groups)**	**Nb.Groups/ Time (0.90 Acc.)**
OPF	**0.885**/4.7s	**0.916**/2.2s	**22**/3.63s
OPF using the local search to find k	**0.885**/1.3s	**0.916**/0.8s	**22**/1.1s
OPF-Large-Data (2000 training samples)	0.873 ± 0.023/0.17s	0.892 ± 0.017/0.13s	32/0.2s
k-Means	0.844 ± 0.025/**0.15*s***	0.877 ± 0.009/**0.12*s***	36/**0.13*s***
OPF-Blocks-2 (6 blocks)	0.87 ± 0.008/0.23s	0.892 ± 0.016/0.2s	33/0.23s

Comparisons in the Protozoans dataset

The experimental results obtained in the Protozoans dataset are shown in Table 5.2. This dataset has 6 classes, so we calculate the accuracy of the clustering methods for 12 and 18 groups — i.e., two and three clusters per protozoa species. OPF is the method with the highest accuracy by far; however, it is also the slowest. k-Means gets the second highest accuracy and is the fastest method for both group quantities. Again, OPF-Large-Data (with 2000 training samples) and OPF-Blocks-2 (with 8 blocks in the first level) get a

similar performance being the first faster. All methods lose class 6 after the true label propagation phase for 12 groups. There are only 89 instances of this class in the data, so finding it is not an easy task. k-Means sometimes loses class 4, too. Only OPF is able to find class 6 for 18 groups. In addition, we compute the number of clusters that each method needs to get an accuracy equal to or greater than 0.87 without losing any class. Again, OPF is the best method only needing 17 clusters, while k-Means is the worst needing 380 groups. OPF-Blocks-2 requires 15 groups less than OPF-Large-Data to achieve the accuracy value mentioned above.

Table 5.2: Experimental results in the Protozoans dataset.

Method	**Accuracy/Time (12 Groups)**	**Accuracy/Time (18 Groups)**	**Nb.Groups/ Time (0.87 Acc.)**
OPF	**0.874**[1]/23.8s	**0.876**/17.3s	**17**/16.9s
OPF using the local search to find k	**0.874**[1]/14.6s	**0.876**/12.3s	**17**/12.3s
OPF-Large-Data (2000 training samples)	0.829 ± 0.032[1]/0.9s	0.835 ± 0.027[1]/0.88s	113/**1.03*s***
k-Means	0.841 ± 0.02[2]/**0.48*s***	0.849 ± 0.008[1]/**0.76*s***	380/8.33s
OPF-Blocks-2 (8 blocks)	0.826 ± 0.03[1]/2.1s	0.837 ± 0.028[1]/2.07s	98/2.08s

[1] The method loses class 6.
[2] The method loses class 6 and sometimes class 4.

Comparisons in the Eggs dataset

Table 5.3 reveals the experimental outcomes in the Eggs dataset. We determine the accuracy of the compared techniques for 9 and 16 groups, assuming almost a cluster per class and two clusters per class, respectively. Once again, OPF gets the best results and remains as the slowest method. OPF loses class 4 for 9 groups, while the others fail to discover classes 2 and 4. Corresponding to 16 groups, only k-Means presents problems to find all egg species (specifically, class 4). k-Means is the fastest method, but it gets the lowest accuracy values by a large margin. OPF-Blocks-2 (with 6 blocks in the first level) and OPF-Large-Data (with 1500 training samples) gets similar results in the experiments. OPF only needs 17 groups to reach an accuracy value of 0.98 without losing any class, while k-Means requires 53 groups. The suggested local search, to find the adjacency parameter k, allows to bring down the OPF execution time from 3.3 to 0.8 seconds for 9 groups.

Table 5.3: Experimental results in the Eggs dataset.

Method	**Accuracy/Time (9 Groups)**	**Accuracy/Time (16 Groups)**	**Nb.Groups/ Time (0.98 Acc.)**
OPF	**0.95**[1]/3.3s	**0.978**/1.1s	**17**/1.06s
OPF using the local search to find k	**0.95**[1]/0.8s	**0.978**/0.8s	**17**/0.9s
OPF-Large-Data (1500 training samples)	0.935 ± 0.02[2]/0.42s	0.97 ± 0.007/0.27s	31/**0.25*s***
k-Means	0.887 ± 0.014[2]/**0.05*s***	0.943 ± 0.01[1]/**0.078*s***	53/0.56s
OPF-Blocks-2 (6 blocks)	0.929 ± 0.02[2]/0.47s	0.967 ± 0.008/0.48s	31/0.438s

[1] The method loses class 4.
[2] The method loses classes 2 and 4.

Comparisons in the RomeSuperpixels dataset

Table 5.4 exhibits the experimental results of the compared clustering methods in the RomeSuperpixels dataset. We evaluate the accuracy of all the techniques for 21 groups, assuming 3 groups per class. We compute the number of groups each method needs to reach an accuracy value of 0.73 without losing any class, too. The OPF technique requires only 31 clusters to reach the accuracy value mentioned above, however, it takes 273 seconds. OPF-Blocks-2 (with 16 blocks in the first level) achieves the best accuracy for 21 groups, but it has some problems finding classes 1 and 6. In addition, the proposed method only requires 60 clusters to avoid losing any class and exceed the accuracy value of 0.73, obtaining the second best result (OPF-Large-Data needs 239 groups for the same objective). OPF-Large-Data (with 2000 training samples) is the fastest method, but it gets the worst accuracy for 21 groups, losing classes 6 and 7 in many iterations. k-Means needs 350 groups and 147 seconds to find all the classes in the data.

Comparisons in the SkinSegmentation dataset

The experimental results obtained in the SkinSegmentation dataset are displayed in Table 5.5. We quantify the accuracy of the compared techniques for 6 groups, assuming 3 groups for class. The original OPF clustering method that executes Algorithm 1 in all data is not feasible for this very large dataset (the dataset has 245,057 samples), so we ignore it in the comparisons. As there are a lot of instances in this dataset, we have two options to run OPF-Blocks-2. The first is to divide the data into approximately 100 blocks and cluster each one by Algorithm 1; and the second one is to divide all the samples into a smaller number of blocks and cluster each one by OPF-Large-Data. As the last option is the fastest, we decide on it. We execute OPF-Blocks-2 with 16 blocks in

Table 5.4: Experimental results in the RomeSuperpixels dataset.

Method	**Accuracy/Time (21 Groups)**	**Nb.Groups/ Time (0.73 Acc.)**
OPF	$0.722/295.6s$	**31**$/273.3s$
OPF using the local search to find k	$0.722/244.9s$	**31**$/240.5s$
OPF-Large-Data (2000 training samples)	0.712 ± 0.05[2]$/$**7.8*s***	$239/$**11.6*s***
k-Means	0.731 ± 0.021[2]$/16.1s$	$350/147.9s$
OPF-Blocks-2 (16 blocks)	**0.735±0.015**[1]$/20.5s$	$60/19.76s$

[1] The method loses class 6 and sometimes class 1.
[2] The method loses classes 6 and 7.

the first level and cluster each one by OPF-Large-Data with 2000 training samples. All methods achieve a similar accuracy value for 6 groups; however, k-Means is the fastest. k-Means only needs 12 groups to achieve an accuracy value of 0.95, while OPF-Blocks-2 requires 33. The execution time of OPF-Large-Data is reduced from 3.04 to 1.3 seconds, for 6 groups, when the local search is used to find the adjacency parameter k.

Table 5.5: Experimental results in the SkinSegmentation dataset.

Method	**Accuracy/Time (6 Groups)**	**Nb.Groups/ Time (0.95 Acc.)**
OPF-Large-Data (2000 training samples)	$0.936 \pm 0.005/3.04s$	$25/1.19s$
OPF-Large-Data using the local search to find k (2000 training samples)	$0.936 \pm 0.004/1.3s$	$25/0.9s$
k-Means	**0.938±0.009/0.15*s***	**12/0.24*s***
OPF-Blocks-2 (16 blocks with 2000 training samples each)	$0.932 \pm 0.009/2.69s$	$33/1.55s$

Comparisons in the LetterRecognition dataset

Table 5.6 presents the experimental outcomes of the compared methods in the LetterRecognition dataset. Once again, OPF is the best method obtaining an accuracy of 0.692 for 156 groups (considering 6 groups for each capital letter). All other methods achieve similar accuracy values for the number of groups mentioned above, although OPF-Large-Data and k-Means lose class 8 (letter 'H') in some iterations. OPF-Large-Data is the fastest method among all. OPF only needs 164 groups to reach an accuracy value of 0.70, while OPF-Blocks-2 requires 468 for the same goal.

Table 5.6: Experimental results in the LetterRecognition dataset.

Method	**Accuracy/Time (156 Groups)**	**Nb.Groups/ Time (0.70 Acc.)**
OPF	**0.692**/2.7s	**164**/2.53s
OPF using the local search to find k	**0.692**/2.15s	**164**/2.08s
OPF-Large-Data (2500 training samples)	0.575 ± 0.01[2]/**0.28*s***	408/**0.26*s***
k-Means	0.58 ± 0.008[1]/0.75s	309/1.2s
OPF-Blocks-2 (8 blocks)	0.582 ± 0.012/0.39s	468/0.57s

[1] The method loses class 8 twice in 50 iterations.
[2] The method loses class 8 four times in 50 iterations.

5.3 Discussion

In this chapter, we evaluate the proposed technique against some baseline methods in two scenarios: image segmentation and the general data clustering problem. Regarding the image segmentation scenario, we compare OPF-Blocks-1 and OPF-Blocks-2 against SLIC, Quick-Shift, a watershed-based method, OPF-Large-Data, and k-Means in three databases of images: GrabCut, Parasites/Impurities, and Liver. In the three databases, the divide-and-conquer extensions get outstanding results. OPF-Blocks-2 outperforms the other methods in all experiments performed in accordance with the superpixel quality metrics used: boundary recall and under-segmentation error. Only k-Means is able to overcome OPF-Blocks-2 in one experiment, according to the boundary recall metric from 100 superpixels in the Liver database. The results of OPF-Blocks-1 are slightly worse than those of OPF-Blocks-2; however, they always exceed the outcomes of OPF-Large-Data, Watershed, Quickshift, SLIC, and many times those of k-Means. Corresponding to the metrics evaluated (accuracy and Dice coefficient) after true label propagation from

the cluster representatives, the divide-and-conquer extensions also get remarkable results. OPF-Blocks-2 is only surpassed by OPF-Large-Data up to 45 clusters in the GrabCut database and by k-Means up to 50 clusters in the Parasites/Impurities database.

We check that grid sampling is a better strategy than random sampling when a reduced pixel sample of the image is needed. We also confirm the need to form a training set with a sufficient number of samples when clustering a large dataset with OPF-Large-Data and that the proposed local search is a good alternative to the exhaustive search when finding the adjacency parameter k. We verify that the number of blocks to divide an image in OPF-Blocks-1 and OPF-Blocks-2 must take into consideration the entropy and size of the image, the consequent execution time of the technique, and the number of desired superpixels in the final segmentation.

We can say that Watershed and SLIC are the fastest techniques, although the OPF-based methods also cluster the images in a reasonable time. Indeed, SLIC, Watershed, OPF-Large-Data, OPF-Blocks-1, and OPF-Blocks-2 are stable methods according to their execution time — i.e., their performance is not influenced by the number of superpixels generated. In contrast, the performance of k-Means and Quick-shift depends on this number.

We present some superpixel segmentations and some binary masks (binary images generated after the true label propagation from the cluster representatives) produced by the compared techniques in the tested databases, highlighting the results of the divide-and-conquer extensions. In most of the images shown, OPF-Blocks-1 and OPF-Blocks-2 (especially, the last) get a good adherence to the objects' boundaries and recover object masks similar to the ground-truth masks.

With regard to the arbitrary data clustering problem, we confirm the superiority of OPF[10] over k-Means in all the tested datasets. The main drawback of OPF is its poor performance (run-time performance) when trying to cluster a large amount of data. The two OPF extensions that deal with this issue (OPF-Large-Data and OPF-Blocks-2) obtain similar results in most of the experiments carried out; however, with OPF-Blocks-2 is more difficult to lose classes in the resulting groups. This must be a consequence of using all the samples, or at least more samples, to train the technique. There is only one dataset in which the experimental results obtained by OPF-Large-Data and OPF-Blocks-2 differ significantly. In the RomeSuperpixels dataset, OPF-Blocks-2 gets an accuracy of 0.735 and needs 60 groups to reach the accuracy value of 0.73 without losing classes, while the corresponding results of OPF-Large-Data are 0.712 and 239 groups, respectively.

[10]We refer here to the technique that executes Algorithm 1 on all samples of the dataset.

Chapter 6

Conclusions

Advances in data acquisition and storage technologies have provided large datasets to support research, technological development, entertainment, medical diagnosis, among others. Due to this huge amount of data, automatic labeling has become an indispensable step in many of these applications. One of the most popular data labeling procedure is *clustering* where the samples are organized into "similar" groups, usually without any domain knowledge. Numerous classes of clustering methods have been proposed in the literature, such as representative-based methods, hierarchical methods, density-based methods, and graph-based methods. However, many of them cannot address large datasets.

In this master thesis, the Optimum-Path Forest framework for the development of pattern recognition techniques is considered. Given an adjacency relation, a path-cost function, and a procedure to estimate prototypes, the OPF algorithm partitions the feature space into an optimum-path forest rooted at those prototypes. Different choices of these parameters lead to clustering and classification approaches, such that a class may be represented by multiple trees and a cluster is represented by a single tree. A sample that belongs to a given tree is said more strongly connected to the root of that tree than to any other root. Therefore, each root propagates its class/cluster label to the samples of its tree.

In order to address the large dataset problem, we presented an extension of the OPF-clustering technique that exploits a divide-and-conquer model with two levels. At the first level, the data is divided into blocks. The samples belonging to each block are clustered separately by the OPF algorithm. Then, the root of each cluster is promoted to the second level. Next, the samples in the second level are clustered with the OPF algorithm again. Finally, the data samples receive the group label of the roots in the second level through their representatives in the first level. We named this method *OPF-Blocks-2*.

When the blocks in *OPF-Blocks-2* contain too many samples, we use a previous variant for large datasets inside each block, the *OPF-Large-Data* algorithm. *OPF-Large-Data* first finds groups in a given training set and then propagates the root labels to the remaining samples by identifying which root would offer an optimum path to it, if the sample were part of the training set. For improvements, we proposed a heuristic search to find the best adjacency parameter k within the interval $[1, k_{\max}]$ which does not affect effectiveness. In the case of image segmentation, we also demonstrated that a reduced training set contains more relevant samples when it is built by grid sampling rather than random sampling.

Our approach, *OPF-Blocks-2*, was evaluated in the image segmentation scenario for different application domains and it outperformed the compared baselines in all experiments. We also assessed *OPF-Blocks-1*, a method that only makes use of the first level to group the pixels and needs a post-processing to merge superpixels generated in different blocks. The merging procedure consists of comparing the color histograms of adjacent superpixels by the Bhattacharyya coefficient. *OPF-Blocks-1* obtained excellent results, but it is not competitive with *OPF-Blocks-2*.

OPF-Block-2 was also evaluated for arbitrary data grouping. It outperformed *OPF-Large-Data* in the RomeSuperpixels dataset and obtained equivalent results in the remaining datasets. It is important to highlight that *OPF-Blocks-2* can retain clusters that represent real classes much better than *OPF-Large-Data*. This must be a consequence of using all samples or at least many samples for training.

Despite the good results achieved by our divide-and-conquer model when clustering large datasets, this proposal has challenges. As can be seen in the last experiments, *OPF-Blocks-2* could not imitate the results of the original OPF-clustering technique that creates the optimal forest with all the samples of the dataset, so there is still much room for improvement. The first issue is the partition of the dataset into blocks in the first level. Each block should contain enough information to produce a valuable partial clustering. In practice, different choices of block size may lead to better results in each dataset. It is also important to explore spatial information when the dataset is an image. A second problem is that the number of clusters in the first level affects the number of samples for grouping in the second level. Different choices for $k_{\max}$ in each block may lead to better results. However, we maintained it fix as though the data entropy in each block were the same.

Therefore, as future work, an upper limit $k_{\max}$ per block, the block sizes and, in the case of images, their spatial locations, should take into account the data entropy inside the blocks in the first level, in order to preserve all natural clusters when grouping samples in the second level. Higher entropy may require lower values of $k_{\max}$ and reduced block sizes. The technique can also be explored in several contexts. In [107], for instance, the authors use *OPF-Large-Data* for active learning. Initially, the user annotates the classes of the cluster roots (which should represent all classes in the dataset). Then, a pattern classifier is trained along the subsequent iterations to select the most informative samples from each cluster for user supervision. First, the use of *OPF-Blocks-2* in this active learning approach may already lead to improvements, especially when the dataset is represented by pixels or superpixels. Second, there are many possible variants of this active learning method. For example, at each iteration, the method selects the same number of samples per group for user supervision. In our case, we have two levels of groups for the selection process and groups with a higher number of classes, according to the classifier, could provide more samples for user supervision. Another application is visual saliency detection [57]. In [76], *OPF-Large-Data* was used to group background pixels near the borders of the image such that the costs of the optimum paths from the cluster roots to the remaining pixels were explored to segment the foreground as a visual saliency. *OPF-Blocks-2* could be used in the place of *OPF-Large-Data* for the same objective. Similarly, we can use our method in the case of brain tissue segmentation

from magnetic resonance images [23]. Note that, the proposed clustering method can also be extended to more than two levels and this may be important in some of the above applications.

Bibliography

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[2] Luis Afonso, Alexandre Vidal, Michelle Kuroda, Alexandre Xavier Falcão, and João P Papa. Learning to classify seismic images with deep optimum-path forest. In *Graphics, Patterns and Images (SIBGRAPI), 2016 29th SIBGRAPI Conference on*, pages 401–407. IEEE, 2016.

[3] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 81–92. VLDB Endowment, 2003.

[4] Charu C Aggarwal and Chandan K Reddy. *Data clustering: algorithms and applications*. CRC press, 2013.

[5] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM, 1998.

[6] Eduardo Barreto Alexandre, Ananda Shankar Chowdhury, Alexandre Xavier Falcão, and Paulo A Vechiatto Miranda. Ift-slic: A general framework for superpixel generation based on simple linear iterative clustering and image foresting transform. In *Graphics, Patterns and Images (SIBGRAPI), 2015 28th SIBGRAPI Conference on*, pages 337–344. IEEE, 2015.

[7] Willian P Amorim, Alexandre X Falcão, João P Papa, and Marcelo H Carvalho. Improving semi-supervised learning through optimum connectivity. *Pattern Recognition*, 60:72–85, 2016.

[8] Willian Paraguassu Amorim, Alexandre Xavier Falcão, and Marcelo Henriques de Carvalho. Semi-supervised pattern classification using optimum-path forest. In *Graphics, Patterns and Images (SIBGRAPI), 2014 27th SIBGRAPI Conference on*, pages 111–118. IEEE, 2014.

[9] Fernanda A Andaló, Paulo AV Miranda, R da S Torres, and Alexandre X Falcão. Shape feature extraction and description based on tensor scale. *Pattern Recognition*, 43(1):26–36, 2010.

[10] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. In *ACM Sigmod record*, volume 28, pages 49–60. ACM, 1999.

[11] Phipps Arabie. Cluster analysis in marketing research. *Advanced methods in marketing research*, pages 160–189, 1994.

[12] Alper Ayvaci and Stefano Soatto. Motion segmentation with occlusions on the superpixel graph. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 727–734. IEEE, 2009.

[13] Francis R Bach and Michael I Jordan. Learning spectral clustering, with application to speech separation. *Journal of Machine Learning Research*, 7(Oct):1963–2001, 2006.

[14] Pierre Baldi and G Wesley Hatfield. *DNA microarrays and gene expression: from experiments to data analysis and modeling*. Cambridge university press, 2002.

[15] Geoffrey H Ball and David J Hall. Isodata, a novel method of data analysis and pattern classification. Technical report, Stanford research inst Menlo Park CA, 1965.

[16] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

[17] Serge Beucher and Fernand Meyer. The morphological approach to segmentation: the watershed transformation. *Optical Engineering-New York-Marcel Dekker Incorporated-*, 34:433–433, 1992.

[18] James C Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media, 2013.

[19] Sanjiv K Bhatia and Jitender S Deogun. Conceptual clustering in information retrieval. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 28(3):427–436, 1998.

[20] Paul S Bradley, Usama M Fayyad, Cory Reina, et al. Scaling clustering algorithms to large databases. In *KDD*, pages 9–15, 1998.

[21] Jeremy Buhler. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, 17(5):419–428, 2001.

[22] Fabio AM Cappabianco, Alexandre X Falcão, and Leonardo M Rocha. Clustering by optimum path forest and its application to automatic gm/wm classification in mr-t1 images of the brain. In *Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008. 5th IEEE International Symposium on*, pages 428–431. IEEE, 2008.

[23] Fábio AM Cappabianco, Alexandre X Falcão, Clarissa L Yasuda, and Jayaram K Udupa. Brain tissue mr-image segmentation via optimum-path forest clustering. *Computer Vision and Image Understanding*, 116(10):1047–1059, 2012.

[24] Jiansheng Chen, Zhengqin Li, and Bo Huang. Linear spectral clustering superpixel. *IEEE Transactions on Image Processing*, 26(7):3317–3330, 2017.

[25] Yizong Cheng. Mean shift, mode seeking, and clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(8):790–799, 1995.

[26] Giovani Chiachia, Aparecido N Marana, João P Papa, and Alexandre X Falcão. Infrared face recognition by optimum-path forest. In *Systems, Signals and Image Processing, 2009. IWSSIP 2009. 16th International Conference on*, pages 1–4. IEEE, 2009.

[27] Krzysztof Chris Ciesielski, Paulo AV Miranda, Alexandre X Falcão, and Jayaram K Udupa. Joint graph cut and relative fuzzy connectedness image segmentation algorithm. *Medical image analysis*, 17(8):1046–1057, 2013.

[28] Dorin Comaniciu. An algorithm for data-driven bandwidth selection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(2):281–288, 2003.

[29] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[30] Kelton AP Costa, Luis AM Pereira, Rodrigo YM Nakamura, Clayton R Pereira, João P Papa, and Alexandre Xavier Falcão. A nature-inspired approach to speed up optimum-path forest clustering and its application to intrusion detection in computer networks. *Information Sciences*, 294:95–108, 2015.

[31] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: a flexible data processing tool. *Communications of the ACM*, 53(1):72–77, 2010.

[32] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[33] Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. ACM, 2001.

[34] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556. ACM, 2004.

[35] Inderjit S Dhillon and Dharmendra S Modha. A data-clustering algorithm on distributed memory multiprocessors. In *Large-scale parallel data mining*, pages 245–260. Springer, 2002.

[36] Chris Ding, Xiaofeng He, and Horst D Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 606–610. SIAM, 2005.

[37] William E Donath and Alan J Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17(5):420–425, 1973.

[38] Richard O Duda, Peter E Hart, and David G Stork. Pattern classification. 2nd. *Edition. New York*, 2001.

[39] Levent Ertöz, Michael Steinbach, and Vipin Kumar. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 47–58. SIAM, 2003.

[40] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[41] Alexandre X Falcão and Felipe PG Bergo. Interactive volume segmentation with differential image foresting transforms. *IEEE Transactions on Medical Imaging*, 23(9):1100–1108, 2004.

[42] Alexandre X Falcão, Bruno S Cunha, and Roberto A Lotufo. Design of connected operators using the image foresting transform. In *Medical Imaging 2001*, pages 468–479. International Society for Optics and Photonics, 2001.

[43] Alexandre X Falcão, Luciano da Fontoura Costa, and BS Da Cunha. Multiscale skeletons by image foresting transform and its application to neuromorphometry. *Pattern recognition*, 35(7):1571–1582, 2002.

[44] Alexandre X Falcão, Jorge Stolfi, and Roberto de Alencar Lotufo. The image foresting transform: Theory, algorithms, and applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(1):19–29, 2004.

[45] Alexandre X Falcão and Jayaram K Udupa. A 3d generalization of user-steered live-wire segmentation. *Medical Image Analysis*, 4(4):389–402, 2000.

[46] Alexandre X Falcão, Jayaram K Udupa, and Flávio K Miyazawa. An ultra-fast user-steered image segmentation paradigm: live wire on the fly. *Medical Imaging, IEEE Transactions on*, 19(1):55–62, 2000.

[47] Alexandre X Falcão, Jayaram K Udupa, Supun Samarasekera, Shoba Sharma, Bruce Elliot Hirsch, and Roberto de A Lotufo. User-steered image segmentation paradigms: Live wire and live lane. *Graphical models and image processing*, 60(4):233–260, 1998.

[48] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

[49] Robson Leonardo Ferreira Cordeiro, Caetano Traina Junior, Agma Juci Machado Traina, Julio López, U Kang, and Christos Faloutsos. Clustering very large multi-dimensional datasets with mapreduce. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 690–698. ACM, 2011.

[50] Douglas H Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine learning*, 2(2):139–172, 1987.

[51] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 670–677. IEEE, 2009.

[52] Sudipto Guha, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams. In *Foundations of computer science, 2000. proceedings. 41st annual symposium on*, pages 359–366. IEEE, 2000.

[53] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: an efficient clustering algorithm for large databases. In *ACM Sigmod Record*, volume 27, pages 73–84. ACM, 1998.

[54] Ivan R Guilherme, Aparecido N Marana, João P Papa, Giovani Chiachia, Luis CS Afonso, Kazuo Miura, Marcus VD Ferreira, and Francisco Torres. Petroleum well drilling monitoring through cutting image analysis and artificial intelligence techniques. *Engineering Applications of Artificial Intelligence*, 24(1):201–207, 2011.

[55] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Is that you? metric learning approaches for face identification. In *Computer Vision, 2009 IEEE 12th international conference on*, pages 498–505. IEEE, 2009.

[56] Alexander Hinneburg, Daniel A Keim, et al. An efficient approach to clustering in large multimedia databases with noise. In *KDD*, volume 98, pages 58–65, 1998.

[57] Xiaodi Hou and Liqing Zhang. Saliency detection: A spectral residual approach. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[58] Alexander I Iliev, Michael S Scordilis, João P Papa, and Alexandre X Falcão. Spoken emotion recognition through optimum-path forest classification using glottal features. *Computer Speech & Language*, 24(3):445–460, 2010.

[59] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[60] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[61] Anil K. Jain, Robert P W Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[62] Eshref Januzaj, Hans-Peter Kriegel, and Martin Pfeifle. Dbdc: Density based distributed clustering. *Advances in Database Technology-EDBT 2004*, pages 529–530, 2004.

[63] George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[64] George Karypis and Vipin Kumar. Multilevelk-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed computing*, 48(1):96–129, 1998.

[65] George Karypis and Vipin Kumar. Parallel multilevel series k-way partitioning scheme for irregular graphs. *Siam Review*, 41(2):278–300, 1999.

[66] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 1990.

[67] Benjamin King. Step-wise clustering procedures. *Journal of the American Statistical Association*, 62(317):86–101, 1967.

[68] Juraj Kostolansky. Sky segmentation using slic superpixels. 2016.

[69] Alex Levinshtein, Adrian Stere, Kiriakos N Kutulakos, David J Fleet, Sven J Dickinson, and Kaleem Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE transactions on pattern analysis and machine intelligence*, 31(12):2290–2297, 2009.

[70] Ming-Yu Liu, Oncel Tuzel, Srikumar Ramalingam, and Rama Chellappa. Entropy rate superpixel segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2097–2104. IEEE, 2011.

[71] Zhi Liu, Xiang Zhang, Shuhua Luo, and Olivier Le Meur. Superpixel-based spatiotemporal saliency detection. *IEEE transactions on circuits and systems for video technology*, 24(9):1522–1540, 2014.

[72] Roberto Lotufo, Alexandrae Falcão, and Franciscao Z Ampirolli. Fast euclidean distance transform using a graph-search algorithm. In *Computer Graphics and Image Processing, 2000. Proceedings XIII Brazilian Symposium on*, pages 269–275. IEEE, 2000.

[73] Roberto Lotufo and Alexandre Falcão. The ordered queue and the optimality of the watershed approaches. In *Mathematical Morphology and its Applications to Image and Signal Processing*, pages 341–350. Springer, 2002.

[74] Roberto A Lotufo, Alexandre X Falcão, and Francisco A Zampirolli. Ift-watershed from gray-scale marker. In *Computer Graphics and Image Processing, 2002. Proceedings. XV Brazilian Symposium on*, pages 146–152. IEEE, 2002.

[75] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[76] Maira Saboia da Silva. Clustering of pixels by image foresting transform and its application in background segmentation of natural images. Master's thesis, Unicamp, 2011.

[77] Lucy AC Mansilla and Paulo AV Miranda. Image segmentation by oriented image foresting transform with geodesic star convexity. In *International Conference on Computer Analysis of Images and Patterns*, pages 572–579. Springer, 2013.

[78] Geoffrey J McLachlan and Kaye E Basford. *Mixture models: Inference and applications to clustering*, volume 84. Marcel Dekker, 1988.

[79] Louis L McQuitty. Elementary linkage analysis for isolating orthogonal and oblique types and typal relevancies. *Educational and Psychological Measurement*, 17(2):207–229, 1957.

[80] Marina Meila and Jianbo Shi. A random walks view of spectral segmentation. 2001.

[81] David Menotti, Giovani Chiachia, Allan Pinto, William Robson Schwartz, Helio Pedrini, Alexandre Xavier Falcão, and Anderson Rocha. Deep representations for iris, face, and fingerprint spoofing detection. *IEEE Transactions on Information Forensics and Security*, 10(4):864–879, 2015.

[82] Paulo AV Miranda, Alexandre Xavier Falcão, and Thiago V Spina. Riverbed: A novel user-steered image segmentation method based on optimum boundary tracking. *IEEE Transactions on Image Processing*, 21(6):3042–3052, 2012.

[83] Paulo AV Miranda and Lucy AC Mansilla. Oriented image foresting transform segmentation by seed competition. *IEEE Transactions on Image Processing*, 23(1):389–398, 2014.

[84] Alastair P Moore, Simon JD Prince, Jonathan Warrell, Umar Mohammed, and Graham Jones. Superpixel lattices. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[85] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

[86] Peer Neubert and Peter Protzel. Superpixel benchmark and comparison. In *Proc. Forum Bildverarbeitung*, pages 1–12, 2012.

[87] Raymond T. Ng and Jiawei Han. Clarans: A method for clustering objects for spatial data mining. *IEEE transactions on knowledge and data engineering*, 14(5):1003–1016, 2002.

[88] Chong-Wah Ngo, Yu-Fei Ma, and Hong-Jiang Zhang. Video summarization and scene detection by graph modeling. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(2):296–305, 2005.

[89] Liadan O'callaghan, Nina Mishra, Adam Meyerson, Sudipto Guha, and Rajeev Motwani. Streaming-data algorithms for high-quality clustering. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 685–694. IEEE, 2002.

[90] João Papa, Alexandre Falcão, Celso Suzuki, and Nelson Mascarenhas. A discrete approach for supervised pattern recognition. *Combinatorial Image Analysis*, pages 136–147, 2008.

[91] João P Papa, Alexandre X Falcão, Victor Hugo C De Albuquerque, and João Manuel RS Tavares. Efficient supervised optimum-path forest classification for large datasets. *Pattern Recognition*, 45(1):512–520, 2012.

[92] João P Papa, Alexandre X Falcão, Alexandre LM Levada, Débora C Corrêa, Denis HP Salvadeo, and Nelson DA Mascarenhas. Fast and accurate holistic face recognition using optimum-path forest. In *Digital Signal Processing, 2009 16th International Conference on*, pages 1–6. IEEE, 2009.

[93] João P Papa, Alexandre X Falcão, Paulo AV Miranda, Celso TN Suzuki, and Nelson DA Mascarenhas. Design of robust pattern classifiers based on optimum-path forests. *Mathematical Morphology and its Applications to Signal and Image Processing (ISMM)*, pages 337–348, 2007.

[94] João P Papa, Alexandre X Falcão, and Celso TN Suzuki. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, 19(2):120–131, 2009.

[95] João Paulo Papa, Alexandre Xavier Falcão, Greice Martins de Freitas, and Ana Maria Heuminski de Avila. Robust pruning of training patterns for optimum-path forest classification applied to satellite-based rainfall occurrence estimation. *IEEE Geoscience and Remote Sensing Letters*, 7(2):396–400, 2010.

[96] João Paulo Papa, Silas Evandro Nachif Fernandes, and Alexandre Xavier Falcão. Optimum-path forest based on k-connectivity: Theory and applications. *Pattern Recognition Letters*, 87:117–126, 2017.

[97] Haim Permuter, Joseph Francos, and Ian Jermyn. A study of gaussian mixture models of color and texture features for image classification and segmentation. *Pattern Recognition*, 39(4):695–706, 2006.

[98] Paulo E Rauber, Alexandre X Falcão, Thiago V Spina, and Pedro J de Rezende. Interactive segmentation by image foresting transform on superpixel graphs. In *Graphics, Patterns and Images (SIBGRAPI), 2013 26th SIBGRAPI-Conference on*, pages 131–138. IEEE, 2013.

[99] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn. Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1-3):19–41, 2000.

[100] Douglas A Reynolds and Richard C Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE transactions on Speech and Audio Processing*, 3(1):72–83, 1995.

[101] Leonardo M Rocha, Alexandre X Falcão, and Luis Geraldo P Meloni. A robust extension of the mean shift algorithm. In *IWCIA Special Track on Applications*, pages 29–38, 2008.

[102] Leonardo Marques Rocha, Fábio AM Cappabianco, and Alexandre Xavier Falcão. Data clustering as an optimum-path forest problem with applications in image analysis. *International Journal of Imaging Systems and Technology*, 19(2):50–68, 2009.

[103] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, pages 309–314. ACM, 2004.

[104] Mehran Sahami and Daphne Koller. *Using machine learning to improve information access*. PhD thesis, Stanford University, Department of Computer Science, 1998.

[105] Priscila Saito, Pedro J de Rezende, Alexandre X Falcão, Celso TN Suzuki, and Jancarlo F Gomes. A data reduction and organization approach for efficient image annotation. In *Proceedings of the 28th annual ACM symposium on applied computing*, pages 53–57. ACM, 2013.

[106] Priscila TM Saito, Pedro J de Rezende, Alexandre X Falcão, Celso TN Suzuki, and Jancarlo F Gomes. An active learning paradigm based on a priori data reduction and organization. *Expert Systems with Applications*, 41(14):6086–6097, 2014.

[107] Priscila TM Saito, Celso TN Suzuki, Jancarlo F Gomes, Pedro J de Rezende, and Alexandre X Falcão. Robust active learning for the diagnosis of parasites. *Pattern Recognition*, 48(11):3572–3583, 2015.

[108] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[109] Jianbing Shen, Yunfan Du, Wenguan Wang, and Xuelong Li. Lazy random walks for superpixel segmentation. *IEEE Transactions on Image Processing*, 23(4):1451–1462, 2014.

[110] Jianbing Shen, Xiaopeng Hao, Zhiyuan Liang, Yu Liu, Wenguan Wang, and Ling Shao. Real-time superpixel segmentation by dbscan clustering algorithm. *IEEE Transactions on Image Processing*, 25(12):5933–5942, 2016.

[111] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[112] Guang Shu, Afshin Dehghan, and Mubarak Shah. Improving an object detector and extracting regions using superpixels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3721–3727, 2013.

[113] Thiago Vallin Spina, Paulo AV de Miranda, and Alexandre Xavier Falcão. Hybrid approaches for interactive image segmentation using the live markers paradigm. *IEEE Transactions on Image Processing*, 23(12):5756–5769, 2014.

[114] Chris Stauffer and W Eric L Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages 246–252. IEEE, 1999.

[115] X Yu Stella and Jianbo Shi. Multiclass spectral clustering. In *null*, page 313. IEEE, 2003.

[116] Celso TN Suzuki, Jancarlo F Gomes, Alexandre X Falcão, João P Papa, and Sumie Hoshino-Shimizu. Automatic segmentation and classification of human intestinal parasites from microscopy images. *IEEE Transactions on Biomedical Engineering*, 60(3):803–812, 2013.

[117] R da S Torres and Alexandre X Falcão. Contour salience descriptors for effective image retrieval and analysis. *Image and Vision Computing*, 25(1):3–13, 2007.

[118] R da S Torres, Alexandre X Falcão, and L da F Costa. A graph-based approach for multiscale shape analysis. *Pattern Recognition*, 37(6):1163–1174, 2004.

[119] R da S Torres, AX Falcão, and L da F Costa. Shape description by image foresting transform. In *Digital Signal Processing, 2002. DSP 2002. 2002 14th International Conference on*, volume 2, pages 1089–1092. IEEE, 2002.

[120] John E Vargas, Priscila TM Saito, Alexandre X Falcão, Pedro J de Rezende, and Jefersson A dos Santos. Superpixel-based interactive classification of very high resolution images. In *Graphics, Patterns and Images (SIBGRAPI), 2014 27th SIBGRAPI Conference on*, pages 173–179. IEEE, 2014.

[121] John Edgar Vargas Muñoz et al. An iterative spanning forest framework for superpixel segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[122] Andrea Vedaldi and Stefano Soatto. Quick shift and kernel methods for mode seeking. *Computer vision–ECCV 2008*, pages 705–718, 2008.

[123] Olga Veksler, Yuri Boykov, and Paria Mehrani. Superpixels and supervoxels in an energy optimization framework. *Computer Vision–ECCV 2010*, pages 211–224, 2010.

[124] Jie Wang and Xiaoqiang Wang. Vcells: Simple and efficient superpixels using edge-weighted centroidal voronoi tessellations. *IEEE Transactions on pattern analysis and machine intelligence*, 34(6):1241–1247, 2012.

[125] Peng Wang, Gang Zeng, Rui Gan, Jingdong Wang, and Hongbin Zha. Structure-sensitive superpixels via geodesic distance. *International journal of computer vision*, 103(1):1–21, 2013.

[126] Wei Wang, Jiong Yang, Richard Muntz, et al. Sting: A statistical information grid approach to spatial data mining. In *VLDB*, volume 97, pages 186–195, 1997.

[127] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[128] CS Warnekar and G Krishna. A heuristic clustering algorithm using union of overlapping pattern-cells. *Pattern Recognition*, 11(2):85–93, 1979.

[129] Wei Wu, Albert YC Chen, Liang Zhao, and Jason J Corso. Brain tumor detection and segmentation in a crf (conditional random fields) framework with pixel-pairwise affinity and superpixel-level features. *International journal of computer assisted radiology and surgery*, 9(2):241–253, 2014.

[130] Zhenyu Wu and Richard Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(11):1101–1113, 1993.

[131] Fan Yang, Huchuan Lu, and Ming-Hsuan Yang. Robust superpixel tracking. *IEEE Transactions on Image Processing*, 23(4):1639–1651, 2014.

[132] Yi Yang, Sam Hallman, Deva Ramanan, and Charless C Fowlkes. Layered object models for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1731–1743, 2012.

[133] Charles T Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on computers*, 100(1):68–86, 1971.

[134] Mohammed J Zaki, Wagner Meira Jr, and Wagner Meira. *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014.

[135] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. In *ACM Sigmod Record*, volume 25, pages 103–114. ACM, 1996.

[136] Weizhong Zhao, Huifang Ma, and Qing He. Parallel k-means clustering based on mapreduce. In *IEEE International Conference on Cloud Computing*, pages 674–679. Springer, 2009.

[137] Chunhui Zhu, Fang Wen, and Jian Sun. A rank-order distance based clustering algorithm for face tagging. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 481–488. IEEE, 2011.

[138] C Lawrence Zitnick and Sing Bing Kang. Stereo for image-based rendering using image over-segmentation. *International Journal of Computer Vision*, 75(1):49–65, 2007.